

Classification de motifs d'intérêt en surveillance de l'environnement

24/01/2024

Rapport final - Projet S7



EQUIPE :
Gabriel ABENHAIM
Nathan ALIMI
Ghiles KEMICHE
Seif ZAAFOURI

CLIENT :
Elisabeth LAHALLE

RÉFÉRENT :
Elisabeth LAHALLE

Abstract

Ce rapport présente une étude sur la classification de motifs d'intérêt dans le cadre de la surveillance de l'environnement en utilisant des techniques de classification basées sur des réseaux de neurones de type ANN et LSTM. Face aux vastes quantités de données générées par les avancées récentes en surveillance environnementale, il devient essentiel d'adopter des méthodes d'analyse sophistiquées. Notre approche combine l'analyse des données et la construction de modèles mathématiques pour générer et identifier des formes d'intérêt. L'analyse inclut également l'étude détaillée du bruit présent dans les données et son impact sur les performances des algorithmes de classification. Les résultats montrent une efficacité notable de ces réseaux de neurones pour classifier les motifs environnementaux, même en présence de bruit, avec des mesures de performance évaluées par des métriques internes et externes. Cette méthode permet d'automatiser et de renforcer le processus de surveillance, améliorant ainsi la précision et l'efficacité des systèmes de surveillance environnementale et permettant d'aider les experts scientifiques dans la prise de décision face à ces problématiques.

Table des matières

1	Introduction	1
2	État de l'art	2
3	Méthodologie	3
3.1	Observation des formes réelles	3
3.2	Génération des formes	4
3.3	Création du dataset	9
3.4	Fonction de coût et métriques d'évaluation	10
3.4.1	Categorical Cross-Entropy	10
3.4.2	Métriques d'évaluation	11
3.5	Architecture des modèles	12
3.5.1	Architecture du Réseau ANN	12
3.5.2	Réseaux de Neurones Récurrents (RNN)	15
3.5.3	Réseaux LSTM (Long Short-Term Memory)	16
3.6	Nos modèles pour la classification	19
3.6.1	Modèles ANN Testés	19
3.6.2	Notre modèle LSTM pour la classification	21
4	Résultats	23
4.1	Artificial Neural Network	23
4.2	Recurrent Neural Network (LSTM)	31
4.3	Comparaisons	37
5	Discussions	38
6	Conclusion	41
	Bibliographie	42

1 Introduction

L'analyse et la classification des données environnementales représentent un défi clé pour la compréhension et la prédiction des phénomènes naturels. Avec la croissance exponentielle des données générées par les systèmes de surveillance modernes, il est devenu crucial d'adopter des méthodes avancées capables d'extraire efficacement des informations pertinentes.

Dans ce contexte, les techniques d'apprentissage profond, et plus spécifiquement les réseaux LSTM (Long Short-Term Memory), ont émergé comme une solution robuste pour traiter les séries temporelles complexes et bruitées. Contrairement aux approches traditionnelles, ces modèles permettent de capturer des dépendances temporelles à long terme, ouvrant de nouvelles perspectives pour la classification des motifs environnementaux.

Les travaux récents, notamment ceux de Hundman *et al.* [1], ont démontré l'efficacité des réseaux LSTM dans des applications telles que la détection d'anomalies dans des systèmes complexes. De même, Géron [2] a mis en avant la capacité des réseaux de neurones profonds à surpasser les méthodes classiques en termes de précision, même en présence de bruit. Inspirée par ces avancées, cette étude vise à évaluer l'utilisation des réseaux LSTM pour la classification automatique des motifs environnementaux.

Ce travail poursuit deux objectifs principaux :

- Explorer et adapter les architectures ANN et LSTM pour identifier et classer précisément différentes formes de données environnementales, même en présence de bruit.
- Analyser l'impact du bruit sur les performances des modèles, en mettant en place des configurations expérimentales variées pour simuler des scénarios réalistes.

Le corps du document est structuré comme suit :

- **Chapitre 2** : Présentation de l'état de l'art, mettant en lumière les progrès récents dans l'analyse des séries temporelles à l'aide de LSTM et d'autres réseaux de neurones.
- **Chapitre 3** : Description détaillée de la méthodologie, incluant la préparation des données, la conception des architectures ANN et LSTM, les techniques d'ajout et de mesure du bruit en enfin la présentation des métriques.
- **Chapitre 4** : Analyse des résultats obtenus, avec un focus sur l'impact des variations de bruit sur la qualité des classifications. Comparaison des modèles ANN et LSTM.
- **Chapitre 5** : Discussion des implications pratiques des résultats pour la surveillance environnementale et les perspectives d'amélioration.
- **Chapitre 6** : Synthèse des contributions principales et propositions pour des recherches futures.

Cette structure vise à guider le lecteur à travers les différents aspects de l'étude, en offrant une vue complète des défis et des solutions proposés pour améliorer la classification des données environnementales à l'aide des réseaux de neurones.

2 État de l'art

L'analyse des motifs environnementaux par des méthodes de classification automatique a considérablement évolué ces dernières années. Historiquement, des techniques comme le clustering K-means étaient couramment utilisées en raison de leur simplicité et de leur efficacité. Cependant, ces méthodes présentaient des limitations, notamment la nécessité de définir un nombre de clusters à l'avance et leur sensibilité aux valeurs initiales. Ces défis ont été partiellement adressés par des approches améliorées, telles que UK-means, qui intègre un terme de pénalité d'entropie pour ajuster les biais [3].

Les techniques d'apprentissage profond, telles que les réseaux LSTM (Long Short-Term Memory), ont révolutionné ce domaine en permettant de mieux capturer les dépendances temporelles dans des données séquentielles. Comme discuté par Hundman dans [1], les réseaux LSTM couplés à des seuils dynamiques non paramétriques ont été utilisés avec succès pour détecter des anomalies dans des systèmes complexes, tels que les engins spatiaux. Cette approche pourrait être directement adaptée à la surveillance environnementale afin d'identifier des anomalies critiques dans des données massives et bruitées).

De plus, Géron [2] souligne que les réseaux de neurones profonds, lorsqu'ils sont correctement configurés, surpassent les méthodes classiques de clustering en termes de précision et de résilience face au bruit. Ces modèles tirent parti de leur capacité à apprendre des représentations riches des données brutes, ce qui les rend particulièrement adaptés aux défis posés par la classification des motifs environnementaux.

Ainsi, bien que des méthodes traditionnelles comme K-means aient joué un rôle important dans les premières phases d'analyse, la transition vers des approches basées sur l'apprentissage profond, notamment avec les LSTM, représente une avancée majeure pour la classification et l'analyse de motifs dans des contextes complexes et dynamiques.

3 Méthodologie

Dans cette étude, l'application d'un algorithme de classification sur les données de séries temporelles ne permet pas de conclure sur la justesse des résultats en l'absence d'une référence claire. Par conséquent, une analyse visuelle et qualitative des séries temporelles est nécessaire. Cette analyse consiste à zoomer sur différentes plages temporelles pour identifier des familles de formes d'intérêt. Ensuite, nous générerons différentes variantes de ces formes en ajustant des paramètres spécifiques tels que la largeur, la hauteur, et la fréquence des motifs. Nous appliquons donc les méthodes de classification, avant de les étendre aux données réelles, sur les données que nous avons générées. Cette approche est particulièrement pertinente car nous avons fixé la longueur des formes dans nos données, ce qui facilite l'évaluation initiale des résultats.

Une fois ces familles de formes identifiées, nous générerons diverses formes pour chaque famille à l'aide de modèles mathématiques simples, permettant de moduler différents paramètres tels que la largeur et la hauteur des pics. En parallèle, différents niveaux de bruit sont ajoutés aux formes générées afin d'étudier leur effet sur la classification. Les données ainsi générées sont ensuite testées avec les modèles de réseaux de neurones ANN et LSTM. La performance de ces algorithmes est évaluée à l'aide de métriques de précision externes. On compare ensuite les résultats obtenus avec les réseaux de neurones et ceux obtenus avec U-K-Means.

Une fois validés, ces algorithmes sont appliqués aux données réelles pour obtenir les résultats finaux. Cette approche permet de s'assurer de la robustesse et de la fiabilité des méthodes de classification utilisées dans cette étude.

3.1 Observation des formes réelles

L'année dernière, une étude des données réelles avait été menée ([4]), afin d'y repérer des formes d'intérêts. Ceci a mené à la définition de 5 motifs différents, 5 classes de courbes dont les définitions vont être rappelées dans la section suivante. On a représenté en Figure 1 quelques formes qui avaient été identifiées.



FIGURE 1 – Formes issues des données réelles

3.2 Génération des formes

La génération de datasets synthétiques joue un rôle fondamental dans l'évaluation et le développement de nos algorithmes d'apprentissage automatique. Ce type de données est particulièrement utile lorsqu'il est difficile ou coûteux de collecter des données réelles. Dans ce rapport, nous détaillons les étapes suivies pour générer un dataset synthétique à partir de formes géométriques mathématiques. Ce processus implique la création de différentes classes de signaux et l'ajout de bruit. Le dataset contient cinq classes de signaux synthétiques qui sont détaillées ci-dessous.

Sauf mention contraire, toutes les mesures de largeur ou de décalage se font en nombre de points. Ex : width=100 signifie que la courbe contient 100 points.

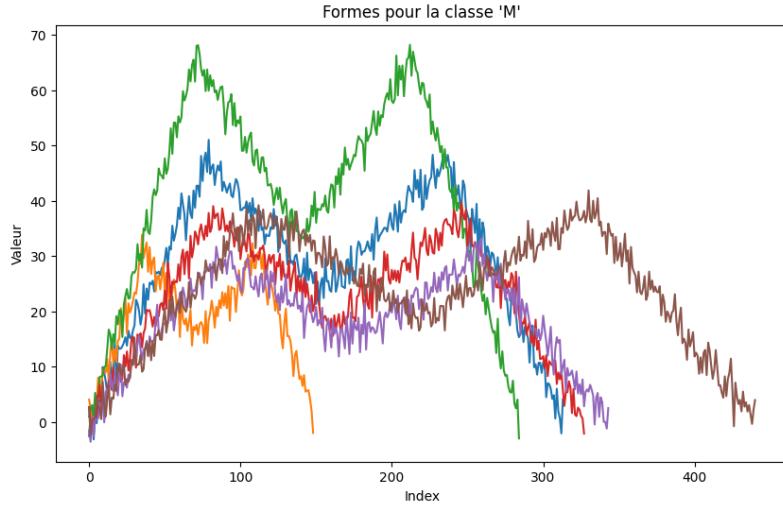
a) Forme "M"

Cette forme est définie par un sommet central élevé entouré de deux vallées à gauche et à droite. Les principaux paramètres sont :

- **Position initiale (s)** : Point de départ de la forme sur l'axe des abscisses.
- **Largeur (w)** : Étendue horizontale de la forme.
- **Hauteur (h)** : Amplitude maximale du signal.

La fonction est définie comme suit :

- Pour i dans $[s, s + w/4[$, $y = \frac{4h \cdot i}{w} - \frac{4h \cdot s}{w}$.
- Pour i dans $[s + w/4, s + w/2[$, $y = -\frac{2h \cdot i}{w} + \frac{h}{2} + \frac{2h \cdot (s+w/2)}{w}$.
- Pour i dans $[s + w/2, s + 3w/4[$, $y = \frac{2h \cdot i}{w} + \frac{h}{2} - \frac{2h \cdot (s+w/2)}{w}$.
- Pour i dans $[s + 3w/4, s + w[$, $y = -\frac{4h \cdot i}{w} + \frac{4h \cdot (s+w)}{w}$.



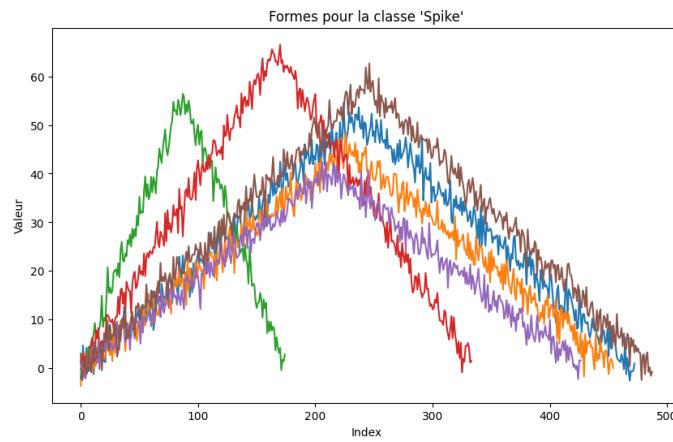
b) Piques ("Spike")

Les signaux en forme de pic sont caractérisés par une montée abrupte suivie d'une descente rapide. Les paramètres incluent :

- **Position initiale (s)** : Point de départ de la forme sur l'axe des abscisses.
- **Largeur (w)** : Durée de l'impulsion.
- **Hauteur (h)** : Amplitude maximale atteinte par le signal.

La fonction est définie comme suit :

- Pour i dans $[0, s[$, $y = 0$.
- Pour i dans $[s, s + w/2[$, $y = \frac{2h \cdot (i-s)}{w}$.
- Pour i dans $[s + w/2, s + w[$, $y = 2h - \frac{2h \cdot (i-s)}{w}$.
- Pour i dans $[s + w, \infty[$, $y = 0$.



c) Parabole ("Parabola")

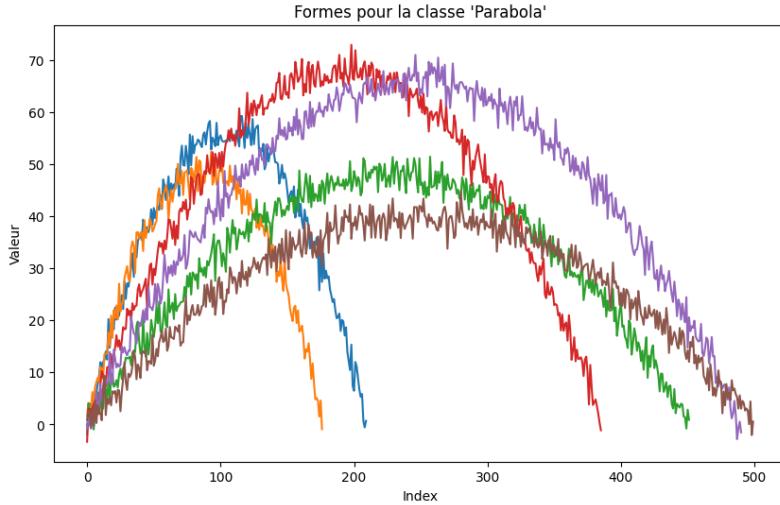
Les signaux en forme de parabole sont caractérisés par une courbe symétrique en forme de U à l'envers. Les paramètres incluent :

- **Position initiale (s)** : Point de départ de la forme sur l'axe des abscisses.

- **Largeur** (w) : Durée (largeur) de la parabole.
- **Hauteur** (h) : Hauteur maximale atteinte par la parabole.

La fonction est définie comme suit :

- Pour i dans $[s, s + w[$, $x = i - s$ et $y = h \cdot \left(1 - \left(\frac{x-w/2}{w/2}\right)^2\right)$.
- Pour i en dehors de cet intervalle, $y = 0$.



d), e) Courbes CLDR et CRDL

Ces formes sont caractérisées par une partie croissante et une partie décroissante. L'une des parties évolue de manière lente (qualifiée de lente grâce à son faible taux de variation), tandis que l'autre partie évolue de manière rapide (qualifiée de rapide grâce à son coefficient directeur élevé). Ces courbes incluent ainsi les *Croissance Lente Décroissance Rapide* (CLDR) et les *Croissance Rapide Décroissance Lente* (CRDL).

Ces signaux ont été modifiés par rapport à l'année dernière afin, d'une part, d'avoir des paramètres en commun avec les autres formes et, d'autre part, de bien marquer une distinction entre CRDL et CLDR.

Les paramètres principaux pour CLDR sont définis comme suit :

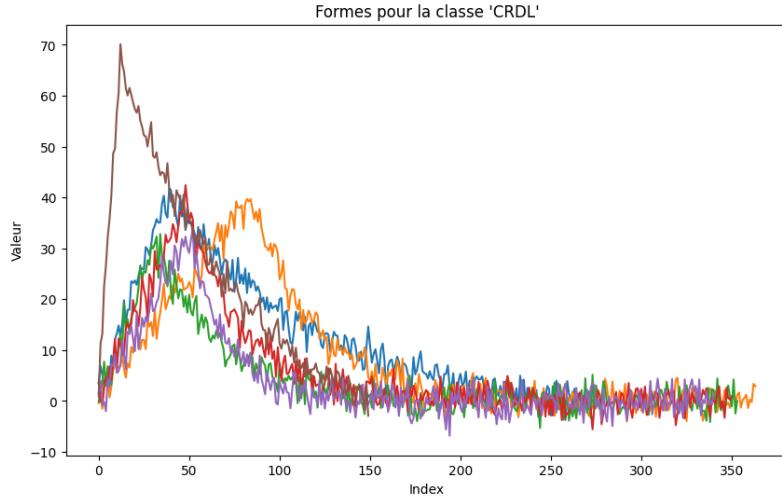
- **Indice de départ** (a) : Indice où commence la courbe (peut être négatif).
- **Largeur** (b) : Largeur totale de la courbe en nombre de points.
- **Ratio de la partie lente** (middle) : Ratio de la largeur de la partie lente par rapport à la largeur totale. Doit être compris entre 0.5 et 1. Le paramètre c utilisé ci-dessous correspond au nombre de points sur la partie lente, et vaut donc $c = \text{middle} \times b$.
- **Amplitude de la partie lente** (A) : Amplitude de la partie lente de la courbe.
- **Hauteur maximale** (h) : Hauteur maximale atteinte par la courbe.

La fonction est définie comme suit :

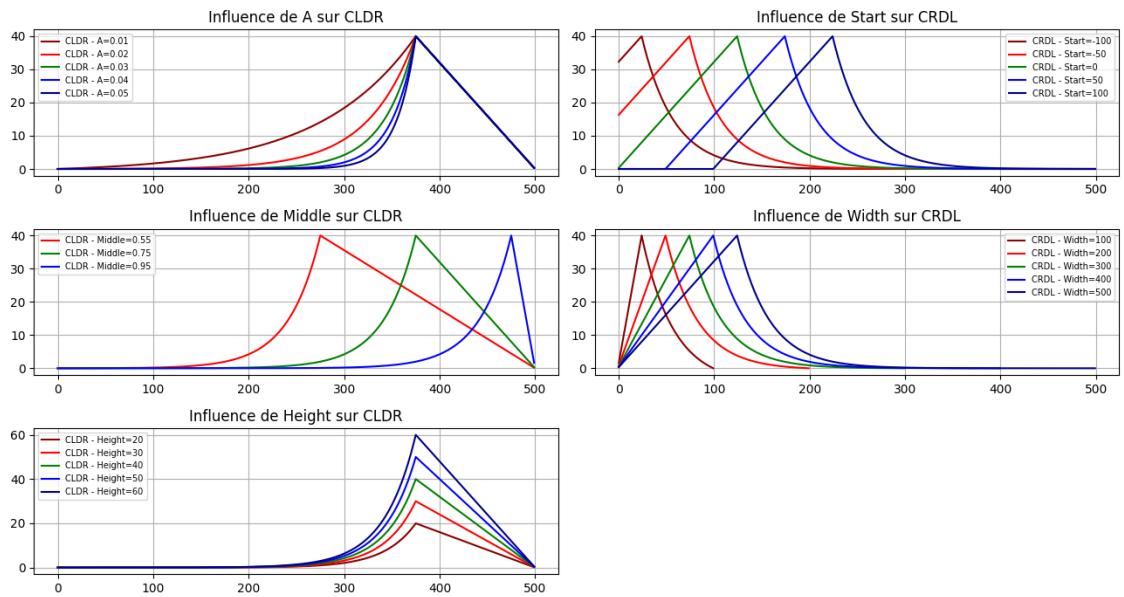
- Pour x dans $[a, c[$: $y = \exp(A \cdot (x - a)) - 1$.

- Pour x dans $[c, b]$, la fonction est linéaire et décroît jusqu'à atteindre 0 au point b . La pente et l'ordonnée à l'origine de cette portion linéaire sont calculées pour assurer que la fonction soit nulle en b et continue en c .

Un CRDL est juste le symétrique d'un CLDR. On peut donc simplement inverser les valeurs de la fonction CLDR, en utilisant $-a$ pour avoir un décalage dans le bon sens.



Afin de tester cette fonction qui a dû être réécrite, nous avons affiché différentes courbes pour voir l'influence de chaque paramètre :



Paramètres et Variations Pour chaque classe de signal, nous introduisons des variations aléatoires des paramètres afin de simuler des données réalistes. La valeur

d'un paramètre $p_{\text{généré}}$ est tirée aléatoirement selon une distribution uniforme :

$$p_{\text{généré}} \sim \text{Uniforme}(p_{\text{base}} - \Delta p, p_{\text{base}} + \Delta p),$$

où p_{base} est la valeur nominale et Δp représente l'amplitude de variation.

Ajout de Bruit

Afin de rendre les signaux plus proches des données réelles, un bruit gaussien est ajouté :

$$x_{\text{bruité}} = x_{\text{original}} + \varepsilon,$$

où $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, et σ est l'écart type de la distribution.

Trois valeurs d'écart type pour le bruit sont utilisés : 0.0 (sans bruit), 2.22 et 5. On a aussi utilisé un dataset avec des niveaux de bruit variés (les 3 niveaux répartis équitablement)

Le choix d'un bruit gaussien est motivé par des observations sur les données réelles des balises de surveillance environnementale. Nous avons identifié des segments où le signal est stationnaire, puis calculé la distribution des variations locales du signal. Cette distribution coïncide avec une distribution gaussienne, avec un écart-type estimé empiriquement par la formule suivante :

$$\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2},$$

Les segments analysés ont été soigneusement choisis pour représenter des périodes où l'activité mesurée reste stable, ce qui rend l'évaluation du bruit plus fiable. Cette approche méthodique assure que le bruit analysé est représentatif des fluctuations aléatoires du signal et non des variations dues à des changements dans les conditions mesurées. Les résultats obtenus sont visualisés dans les histogrammes suivants (Figure 2), avec les ajustements de distributions normales pour chaque période analysée.

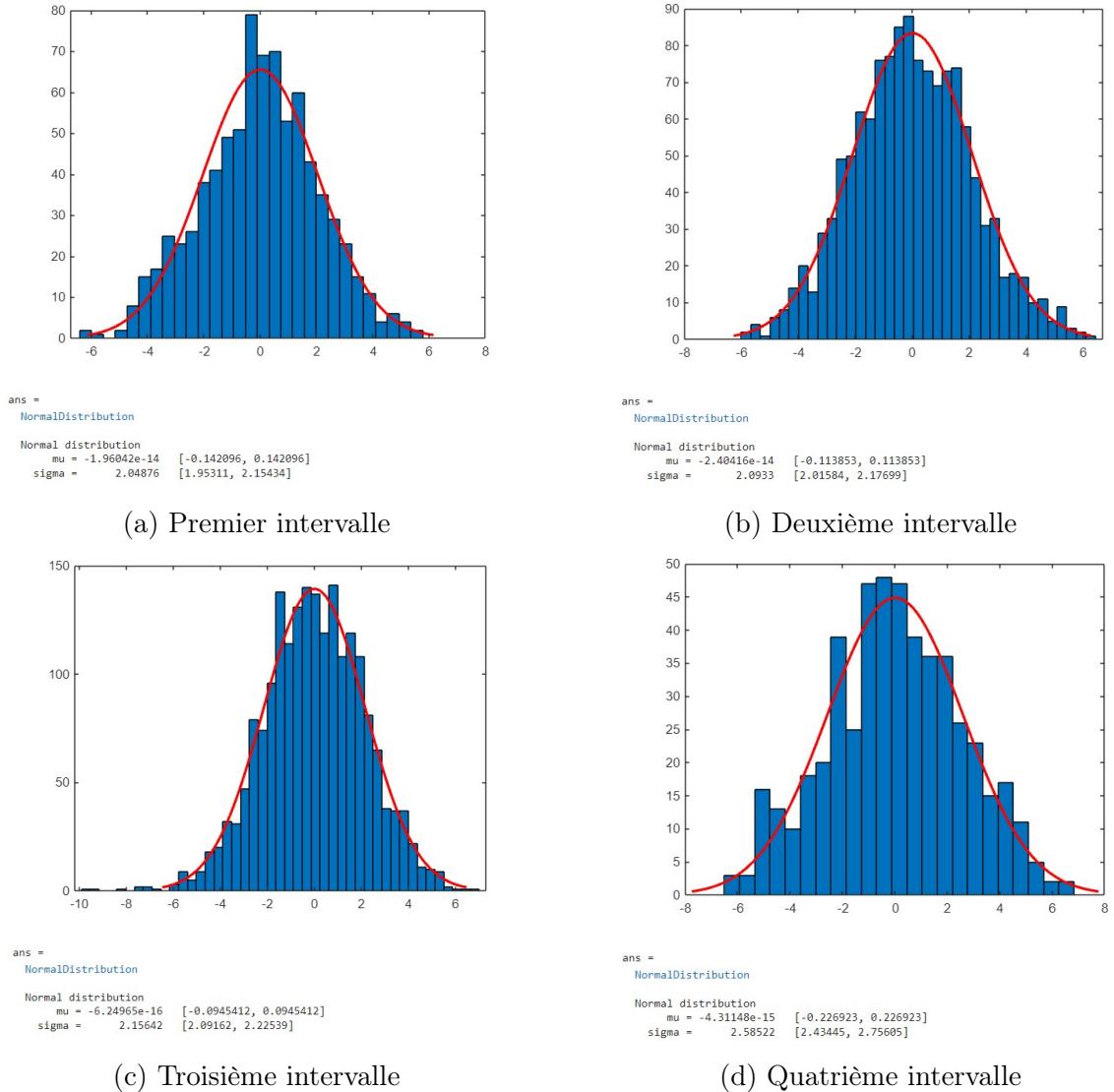


FIGURE 2 – Analyse de la distribution du bruit pour différents intervalles de temps

L'écart-type empirique obtenu est $\hat{\sigma} \approx 2,22$, ce qui a guidé notre choix des valeurs d'écart-types utilisées pour le bruit gaussien ajouté.

3.3 Création du dataset

On a créé une classe Dataset de formes différentes pour entraîner nos modèles. Les formes incluent un nombre équilibré de formes citées précédemment. Les paramètres principaux sont :

- **Nombre de formes par classe** (*formsPerClass*) : Nombre de formes générées pour chaque type de signal.
- **Niveaux de bruit** (*noise_levels*) : Liste des niveaux de bruit ajoutés aux formes.
- **Largeurs des formes** (*widths_list*) : Liste des largeurs des formes générées.

- **Variation de largeur** (*width_variation*) : Liste des variations de largeur pour chaque forme.

Paramètre	Description	Valeur par défaut
<i>formsPerClass</i>	Nombre de formes par classe	500
<i>noise_levels</i>	Niveaux de bruit	[0.0, 2.22, 5.0]
<i>widths_list</i>	Largeurs des formes	[500]
<i>width_variation</i>	Variation de largeur	[200]

TABLE 1 – Paramètres de génération de dataset

Les principales étapes du processus sont :

1. **Initialisation** : Création d'une instance de la classe *DatasetLSTM* avec les paramètres spécifiés.
2. **Génération des formes** : Utilisation de fonctions spécifiques pour générer différentes formes avec des paramètres aléatoires et ajout de bruit.
3. **Création du DataFrame** : Compilation des formes générées dans un DataFrame avec leurs métadonnées.
4. **Padding des données** : Complétion des formes avec des 0 pour qu'elles aient toutes la même longueur. *Remarque : les 0 ajoutés seront ensuite ignorés par le LSTM grâce à une couche de Masking, voir section 3.5.3.*
5. **Encodage et normalisation** : Encodage des labels en one-hot et normalisation des données.
6. **Division du dataset** : Séparation du dataset en ensembles d'entraînement (70%), de validation (15%) et de test (15%).

3.4 Fonction de coût et métriques d'évaluation

3.4.1 Categorical Cross-Entropy

La *categorical cross-entropy* est la fonction de perte la plus communément utilisée pour les tâches de classification multi-classes en deep learning. Également appelée *entropie croisée catégorielle*, elle mesure la distance entre la distribution prédite par le réseau et la distribution réelle des classes.

Définition mathématique. Soient y la vraie étiquette d'un exemple, représentée sous forme *one-hot* (matrice de dimension $(1 \times C)$ pour C classes) et \hat{y} la distribution de probabilités prédite par le réseau de neurones (également $(1 \times C)$). Alors la fonction de perte \mathcal{L} (CCE) s'écrit :

$$\begin{array}{ccc}
 \text{Prédiction du modèle } \hat{\mathbf{y}} & & \text{Vecteur One-Hot } \mathbf{y} \\
 \boxed{\hat{\mathbf{y}} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.0 \\ 0.1 \\ 0.0 \end{bmatrix}} & \xrightarrow{\mathcal{L}(y, \hat{y}) = - \sum_{i=1}^5 y_i \log(\hat{y}_i)} & \boxed{\mathbf{y} = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}}
 \end{array}$$

où

- $y_i = 1$ si l'exemple appartient à la classe i , et 0 sinon (*encodage one-hot*),
- \hat{y}_i est la probabilité prédictive que l'exemple appartienne à la classe c , typiquement donnée par la fonction d'activation *softmax*.

Interprétation.

- Lorsque la prédiction est parfaite, $\hat{y}_c = 1$ pour la bonne classe c , ce qui rend $\ln(\hat{y}_c) = \ln(1) = 0$. La perte est alors $\mathcal{L} = 0$.
- En revanche, plus la probabilité prédictive pour la classe correcte s'éloigne de 1, plus la valeur $-\ln(\hat{y}_c)$ grandit, augmentant la perte.

Implémentation pratique. Avec la bibliothèque *Keras*, cette fonction de perte est directement disponible sous le nom `categorical_crossentropy`. Nous avons ainsi utilisé :

- Une couche de sortie contenant $C = 5$ neurones.
- Une activation *softmax* sur cette couche.
- Et la perte CCE sur ces prédictions probabilistes.

Son utilisation garantit une interprétation claire de la sortie du réseau et un ajustement progressif des poids pour minimiser la différence entre la distribution prédictive et la distribution réelle.

3.4.2 Métriques d'évaluation

Dans le cadre de l'évaluation de la performance de nos réseaux de neurones (*ANN* et *LSTM*) dédiés à la classification, il est essentiel de disposer d'indicateurs permettant de mesurer de façon fiable la correspondance entre les prédictions du modèle et les étiquettes réelles. Les métriques présentées ci-dessous sont couramment utilisées pour quantifier la qualité de la classification et faciliter la comparaison entre différents modèles ou différents paramétrages.

— Accuracy (Exactitude)

L'accuracy mesure la proportion de prédictions correctes parmi l'ensemble des prédictions effectuées. Elle est souvent considérée comme l'indicateur le plus intuitif de la performance globale :

$$\text{Accuracy} = \frac{\text{nombre de prédictions correctes}}{\text{nombre total de prédictions}}$$

— **Précision (Precision)**

La précision évalue la capacité du modèle à ne prédire positif que lorsque la classe est effectivement positive. Elle est définie comme suit :

$$\text{Précision} = \frac{TP}{TP + FP}$$

où :

- *TP* (True Positives) représente le nombre d'exemples correctement prévus positifs,
- *FP* (False Positives) représente le nombre d'exemples incorrectement prévus positifs.

— **Rappel (Recall) ou Sensibilité**

Le rappel mesure la proportion d'exemples correctement identifiés comme positifs parmi ceux qui le sont réellement. Il se calcule comme suit :

$$\text{Rappel} = \frac{TP}{TP + FN}$$

où *FN* (False Negatives) représente le nombre d'exemples qui sont positifs en réalité mais prédits négatifs par le modèle.

— **Score F1 (F1 Score)**

Le score F1 est la moyenne harmonique de la précision et du rappel. Il prend en compte à la fois les faux positifs et les faux négatifs, et se révèle particulièrement utile lorsque les classes sont déséquilibrées :

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

3.5 Architecture des modèles

3.5.1 Architecture du Réseau ANN

Un réseau de neurones artificiels (*Artificial Neural Network*, ANN) est un modèle inspiré du fonctionnement biologique des neurones, conçu pour résoudre divers problèmes d'apprentissage, y compris la classification. Voici une description approfondie des éléments et étapes impliqués dans sa structure et son fonctionnement :

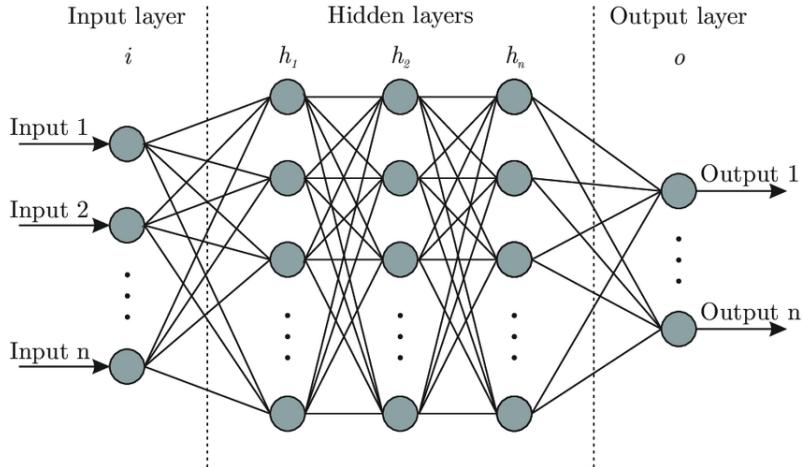


FIGURE 3 – Architecture d'un réseau de neurone ANN

1. Structure du réseau

1.1 Couche d'entrée :

— **Rôle** : Reçoit les données brutes ou les caractéristiques extraites.

1.2 Couche cachée entièrement connectée :

- **Composition** :
- Chaque couche ℓ est formée d'un ensemble de n_ℓ neurones, où chaque neurone applique une transformation affine suivie d'une fonction d'activation non linéaire.
- Transformation mathématique :

$$z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)} \in \mathbb{R}^{n_\ell}$$

où :

- $W^{(\ell)}$ est la matrice des poids pour la couche ℓ , donc $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$
- $b^{(\ell)} \in \mathbb{R}^{n_\ell}$ est le vecteur de biais,
- $a^{(\ell-1)} \in \mathbb{R}^{n_{\ell-1}}$ représente les sorties activées de la couche précédente (qui a $\ell - 1$ neurones).

1.3 Couche de sortie :

- **Rôle** : Fournit une probabilité pour chaque classe.

- **Fonction d'activation :**

- Utilisation de la fonction softmax pour convertir les scores en probabilités :

$$P(\hat{y}^{(i)} = c \mid x^{(i)}) = \frac{\exp(z_c)}{\sum_{k=1}^K \exp(z_k)}$$

où

- $x^{(i)}$ est la i -ème série temporelle
- $\hat{y}^{(i)}$ est la prédiction du modèle pour l'entrée $x^{(i)}$
- z_c est le score brut de la classe c
- K est le nombre total de classes

- **Sortie :** Un vecteur de probabilités $\hat{y}^{(i)}$ dont la somme des composantes est égale à 1. Chaque composante $\hat{y}_k^{(i)}$ est la probabilité prédictive que $x^{(i)}$ appartienne à la classe k .

2. Fonctionnement

2.1 Initialisation des paramètres :

- Les poids W sont initialisés à de petites valeurs aléatoires
- Les biais b sont initialisés à zéro ou à de petites valeurs constantes.

2.2 Propagation avant (Forward Propagation) :

- Les données passent de la couche d'entrée à la couche de sortie via les transformations successives dans chaque couche.
- À chaque étape :
 - Calcul des activations intermédiaires $a^{(\ell)}$.
 - Calcul des scores finaux $z^{(L)}$ (dernière couche).

2.3 Fonction de perte :

- Perte d'entropie croisée pour les tâches de classification :

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_k^{(i)} \log P(\hat{y}^{(i)} = c \mid x^{(i)})$$

où :

- N est le nombre d'exemples,
- K est le nombre total de classes
- $y_k^{(i)}$ est une variable indicatrice (1 si l'exemple i appartient à la classe k , 0 sinon).
- $P(\hat{y}^{(i)} = c \mid x^{(i)})$ est la sortie de la couche softmax.

2.4 Optimisation :

— Algorithme :

- L'algorithme de rétropropagation est utilisé pour calculer les gradients de la perte par rapport aux paramètres $W^{(\ell)}$ et $b^{(\ell)}$.
- Ces gradients sont calculés pour chaque couche en partant de la sortie vers l'entrée (rétrôpropagation des erreurs).
- Mise à jour des paramètres :

- Utilisation d'un optimiseur comme l'algorithme **Adam** ou la descente de gradient stochastique (SGD) :

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$$

où η est le taux d'apprentissage, et θ représente les paramètres.

3. Hyperparamètres clés

- **Nombre de couches cachées** : Typiquement entre 2 et 5 pour notre cas.
- **Nombre de neurones par couche** : Dépend de la complexité des données, souvent 2^n (par exemple, 32, 64, 128, 256, etc.).
- **Taux d'apprentissage (η)** : Ajusté pour garantir la convergence.
- **Techniques de régularisation** :
 - **Dropout** : Désactiver aléatoirement un pourcentage de neurones.
 - **L2 regularization** : Ajouter une pénalité proportionnelle au carré des poids.

Ce modèle est bien adapté comme point de départ pour des tâches où les motifs temporels ne nécessitent pas une mémorisation explicite à long terme (comparé aux LSTM). Pour une classification robuste, les ANN peuvent être utilisés en parallèle avec des techniques d'extraction avancée de caractéristiques.

3.5.2 Réseaux de Neurones Récursifs (RNN)

Les réseaux de neurones récurrents (RNN) sont une classe de réseaux de neurones conçus pour traiter les données séquentielles, telles que les séries temporelles, le texte ou les signaux audio. Contrairement aux réseaux neuronaux classiques, les RNN possèdent des connexions récurrentes qui leur permettent de prendre en compte les informations des états précédents.

Représentation d'un RNN :

Un RNN peut être visualisé comme un "réseau déplié" (voir Figure 4) sur une séquence temporelle. à chaque étape t , le réseau produit un état caché h_t en fonction de l'entrée actuelle x_t et de l'état caché précédent h_{t-1} :

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h),$$

où W_h , W_x , et b_h sont les poids et le biais.

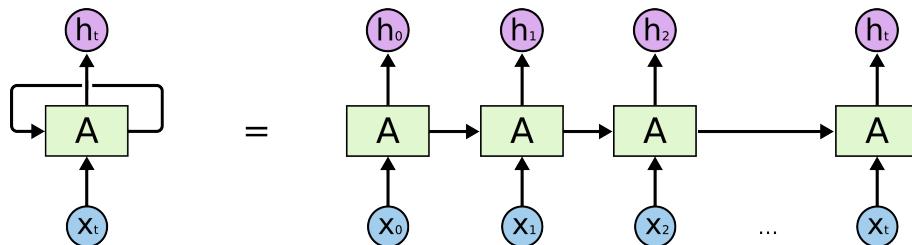


FIGURE 4 – RNN *déroulé* dans le temps. Chaque bloc répète la même structure et partage les mêmes poids.

Limitations des RNN classiques

Malgré leur capacité à modéliser des séquences, les RNN classiques présentent plusieurs problèmes majeurs :

- **Problème de gradient** : Les RNN souffrent souvent de l'atténuation ou de l'explosion du gradient lors de l'apprentissage, ce qui rend difficile la prise en compte des dépendances à long terme.
- **Mémoire limitée** : Les RNN classiques ont une capacité limitée à retenir les informations sur de longues périodes.
- **Difficulté à traiter des séquences longues** : Les performances des RNN se dégradent sur des séquences très longues.

3.5.3 Réseaux LSTM (Long Short-Term Memory)

Pour résoudre ces problèmes, les réseaux LSTM ont été conçus comme une extension des RNN, comme mentionné par Géron dans [2]. Les LSTM introduisent un mécanisme de "mémoire", qui est contrôlé par des **portes**, qui régulent le flux d'informations à travers la cellule. Une cellule LSTM comporte trois portes principales :

- **Porte d'oubli** : Décide quelles informations doivent être "oubliées" de l'état de la cellule.
- **Porte d'entrée** : Détermine quelles nouvelles informations doivent être ajoutées à l'état de la cellule.
- **Porte de sortie** : Contrôle quelles informations de l'état de la cellule doivent être utilisées pour générer la sortie.

L'architecture d'une cellule LSTM est montrée en Figure 5.

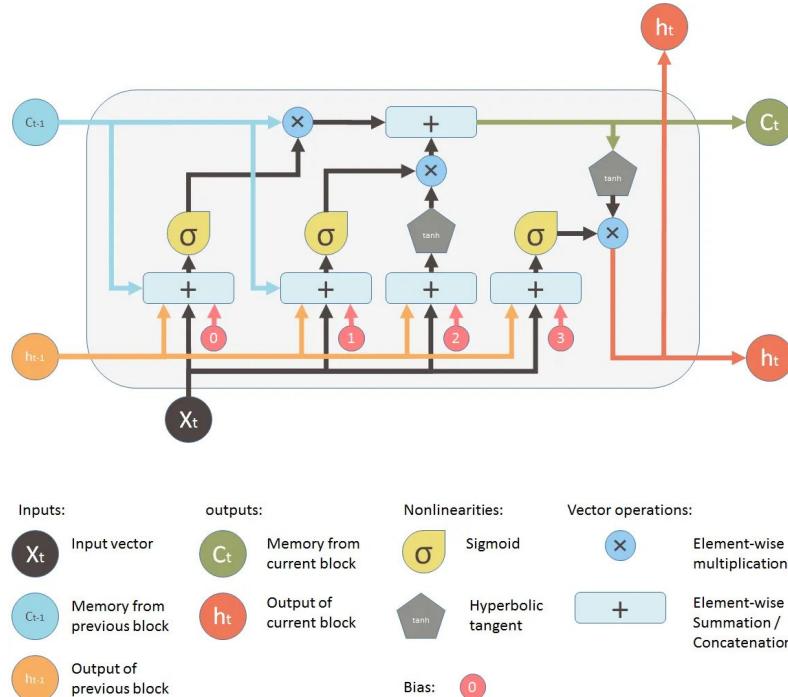


FIGURE 5 – Architecture d'une cellule LSTM

Pour traiter une série temporelle (x_1, x_2, \dots, x_T) , la cellule LSTM est *déroulée* dans le temps. On l'applique successivement sur chaque x_t , et on propage l'état caché h_t et l'état de la mémoire C_t .

Comme illustré en Figure 6, la cellule LSTM est répliquée à chaque instant, partageant les mêmes paramètres, et reçoit comme entrées (x_t, h_{t-1}, C_{t-1}) pour produire (h_t, C_t) . Cette structure permet de propager l'information et les gradients de manière efficace sur de longues séquences.

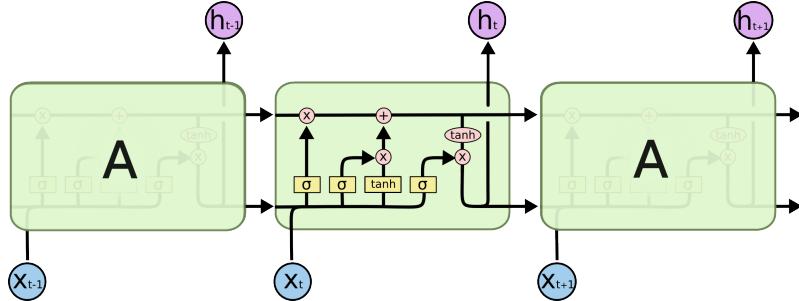


FIGURE 6 – Réseau LSTM *déroulé* dans le temps

Le comportement d'une cellule LSTM est ainsi décrit par les équations suivantes :

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (\text{Porte d'oubli}) \\ i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (\text{Porte d'entrée}) \\ \tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \quad (\text{Valeurs candidates}) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (\text{Mise à jour de l'état de la cellule}) \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (\text{Porte de sortie}) \\ h_t &= o_t \odot \tanh(C_t) \quad (\text{état caché}) \end{aligned}$$

Avantages des LSTM

- **Résolution des problèmes de gradient** : Les LSTM gèrent efficacement les problèmes de gradient, permettant d'apprendre des dépendances à long terme.
- **Mémoire adaptative** : Les portes permettent de contrôler dynamiquement quelles informations conserver ou oublier.
- **Applications diverses** : Les LSTM sont largement utilisés dans la reconnaissance vocale, le traitement du langage naturel et la prédiction de séries temporelles.

Ajout d'une couche de Masquage (ou *Masking layer*)

Dans les modèles traitant des données séquentielles, il est souvent nécessaire de standardiser la longueur des séquences d'entrée, en ajoutant des valeurs spéciales (par exemple, des zéros) pour combler les séquences plus courtes. Ces valeurs ajoutées n'ont pas de signification réelle pour le modèle et ne doivent pas être prises en

compte lors des calculs du LSTM déroulé. Pour résoudre ce problème, une couche de **Masquage** est ajoutée au modèle. Elle identifie les positions des données à ignorer dans chaque séquence d'entrée et applique un masque pour empêcher ces valeurs d'influencer les calculs au sein du LSTM déroulé. Ainsi, seules les données pertinentes contribuent à l'état de la cellule et à la sortie du modèle, améliorant ainsi sa précision et sa robustesse.

En pratique, cette couche est essentielle pour garantir que les LSTM se concentrent uniquement sur les informations significatives.

3.6 Nos modèles pour la classification

3.6.1 Modèles ANN Testés

Dans cette étude, plusieurs architectures de réseaux de neurones artificiels (ANN) ont été testées afin d'explorer leurs performances sur des configurations variées de données. Chaque modèle a été conçu pour répondre à une problématique spécifique, comme l'impact de la profondeur, du dropout, ou de la régularisation L_2 . Ces choix permettent d'évaluer les compromis entre la complexité du modèle, sa capacité de généralisation et sa robustesse face au bruit ou aux données partielles.

- **Simple Model** : Une architecture de base avec une seule couche cachée. Ce modèle sert de référence pour évaluer la complexité minimale requise pour résoudre le problème.
- **Deep Model** : Deux couches cachées sont introduites pour examiner l'impact de la profondeur sur la capacité de représentation.
- **Dropout Model** : Le dropout est utilisé (20% par couche) pour limiter le surapprentissage et renforcer la robustesse du modèle.
- **L2 Regularized Model** : Une régularisation L_2 est appliquée aux couches pour prévenir une complexité excessive, particulièrement utile pour les données bruitées.
- **Combined Model** : Ce modèle combine la régularisation L_2 et le dropout pour bénéficier des avantages des deux techniques.
- **Mix L2 Regularized and Dropout Model** : Une variante dans laquelle les techniques de régularisation et de dropout sont appliquées de manière ciblée à différentes couches.
- **Intense Learning Model** : Ce modèle utilise un faible taux d'apprentissage (10^{-5}) et un entraînement prolongé (70 époques) pour traiter des données complexes.
- **Deep and Intense Model** : Une architecture encore plus profonde avec un entraînement sur 100 époques pour explorer les capacités d'apprentissage profond sur des données variées.

La table 7 résume les configurations des modèles testés.

Model Name	Layers	Dropout	Regularization	L-Rate	Epochs	Batch
Simple Model	Layer-1 (64)	None	None	0.001	10	8
Deep Model	Layer-1 (32), Layer-2 (64)	None, None	None, None	0.0001	10	8
Dropout Model	Layer-1 (64), Layer-2 (32)	0.2, 0.2	None, None	0.001	20	8
L2 Regularized Model	Layer-1 (128), Layer-2 (64)	None, None	l2(0.01), l2(0.01)	0.001	10	8
Combined Model	Layer-1 (128), Layer-2 (64), Layer-3 (32)	None, 0.3, None	l2(0.01), None, None	0.001	10	8
Mix L2 Regularized and Dropout Model	Layer-1 (128), Layer-2 (64), Layer-3 (32)	0.1, None, 0.1	None, l2(0.01), None	0.001	10	8
Intense learning Model	Layer-1 (128), Layer-2 (64), Layer-3 (32)	0.1, 0.1, 0.1	None, None, None	1e-05	70	8
Deep and Intense Model	Layer-1 (128), Layer-2 (64), Layer-3 (32)	0.1, 0.1, 0.1	None, None, None	1e-05	100	8

FIGURE 7 – Configurations des modèles ANN testés. Chaque modèle explore une dimension spécifique du design de réseau (profondeur, dropout, régularisation, etc.).

Modèle Optimal

Après de nombreux essais et évaluations sur différentes configurations de données (bruit, largeurs variées, formes complètes/incomplètes, etc.), nous avons identifié le modèle qui offre des performances robustes sur la plupart des scénarios testés. Ce modèle est défini par les caractéristiques suivantes :

Layer (type)	Output Shape	Param #
Hidden Layer 1 (Dense)	(None, 128)	32,512
Dropout 1 (0.1) (Dropout)	(None, 128)	0
Hidden Layer 2 (Dense)	(None, 64)	8,256
Dropout 2 (0.1) (Dropout)	(None, 64)	0
Hidden Layer 3 (Dense)	(None, 32)	2,080
Dropout 3 (0.1) (Dropout)	(None, 32)	0
Output Layer (5 units) (Dense)	(None, 5)	165

FIGURE 8 – Sommaire du modèle ANN Optimal

- **Architecture** : Le modèle comprend trois couches cachées avec des tailles respectives de 128, 64, et 32 neurones. (Voir le schéma dans la Figure 9 ci-dessous)
- **Dropout** : Un taux de dropout de 0.1 est appliqué après chaque couche cachée pour éviter le surapprentissage.
- **Régularisation** : Aucune régularisation explicite (L2 ou L1) n'a été nécessaire, car le Dropout s'est avéré suffisant pour stabiliser l'apprentissage.
- **Taux d'apprentissage** : Le modèle utilise un taux d'apprentissage faible de 0.00001, garantissant une convergence progressive et minimisant les oscillations dans la courbe de perte.
- **Nombre d'époques** : Le modèle est entraîné sur 100 époques pour permettre une exploration approfondie des données.
- **Taille de batch** : Une taille de batch de 8 a été utilisée, offrant un compromis entre stabilité et vitesse d'entraînement.

Ce modèle a démontré une excellente capacité à généraliser sur des données bruitées, tronquées, et de largeurs variées.

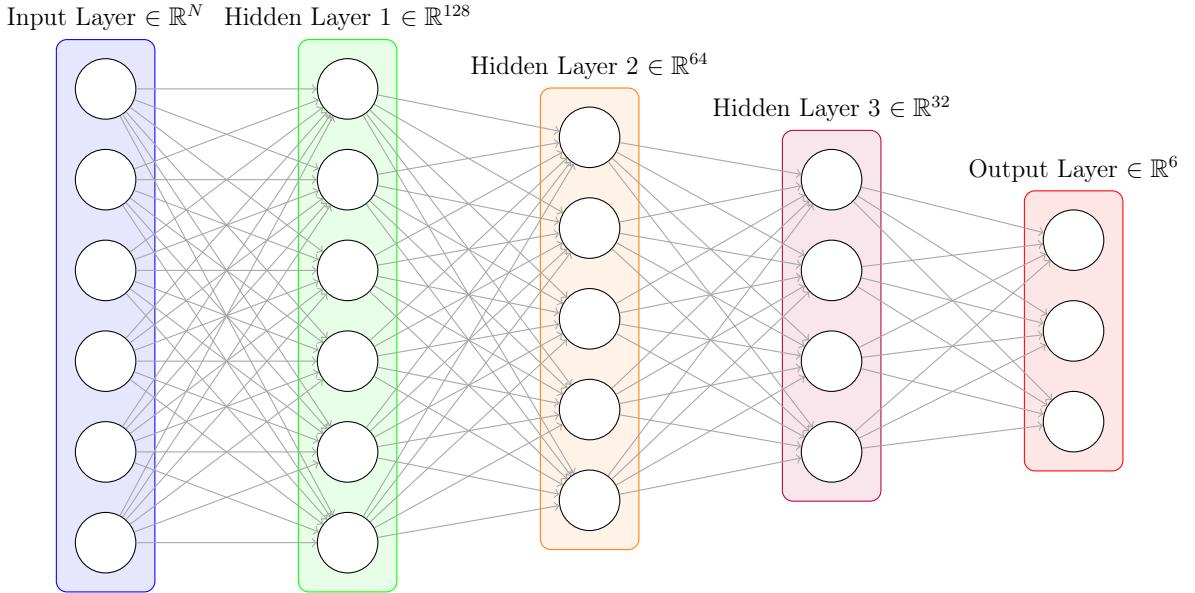


FIGURE 9 – Schéma du modèle ANN optimal

Justification du Choix des Paramètres

- **Dropout** : Des taux de 0.1 ont été choisis après comparaison avec des valeurs plus élevées (0.2, 0.3), qui ont montré une légère sous-performance, particulièrement sur des données bruitées.
- **Taux d'apprentissage** : L'utilisation d'un learning rate faible a permis d'éviter les oscillations dans les configurations complexes (données bruitées et formes incomplètes), où des taux plus élevés (ex. 0.001) menaient à une instabilité.
- **Nombre d'époques** : 100 époques se sont avérées suffisantes pour la convergence sans surapprentissage.

Ce modèle optimal constitue une référence robuste pour la suite des expérimentations et peut servir de base pour des applications pratiques sur des données similaires.

3.6.2 Notre modèle LSTM pour la classification

Pour le réseau LSTM, on a opté pour l'architecture suivante avec une couche de cellules LSTM et un Dropout pour améliorer la précision du modèle (et la couche de Masking, dont l'utilité a été discutée plus haut).

Layer (type)	Output Shape	Param #
Masking_Layer (Masking)	(None, 500, 1)	0
LSTM (LSTM)	(None, 64)	16,896
Dropout_Layer_1 (Dropout)	(None, 64)	0
Output_Layer (Dense)	(None, 5)	325

FIGURE 10 – Architecture du réseau LSTM utilisé

D'autres tests ont également été effectués avec des LSTM bidirectionnels (un LSTM bidirectionnel traite les séquences en passant les données dans deux directions : une vers l'avant et une vers l'arrière, ce qui permet au modèle de prendre en compte à la fois les dépendances passées et futures dans la séquence), mais cette structure était plus longue à entraîner et n'apportait pas de réel gain de performance.

4 Résultats

4.1 Artificial Neural Network

Résultats sur des formes complètes de même largeur et sans bruit

Après l'entraînement du **modèle simple** présenté dans 3.6.1 sur des données non bruitées, les résultats suivants ont été obtenus :

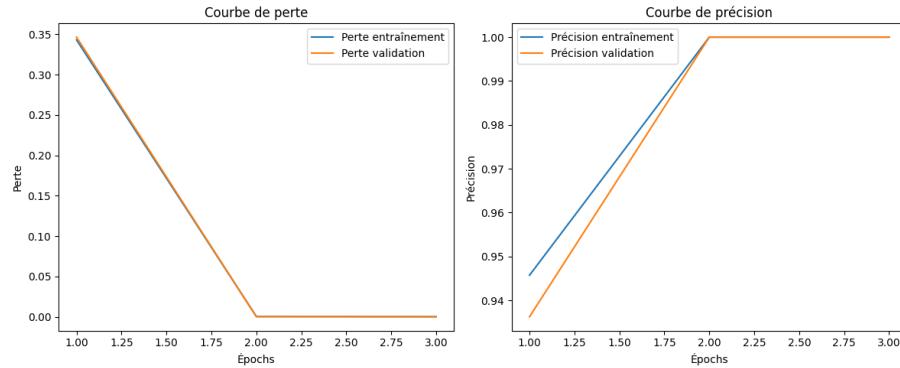


FIGURE 11 – Courbes de perte et de précision pour l'entraînement et la validation.

Comme illustré dans la figure 11, le modèle converge rapidement et de manière stable. La perte diminue régulièrement et la précision atteint 100% dès les premières époques, reflétant la simplicité du problème en l'absence de bruit.

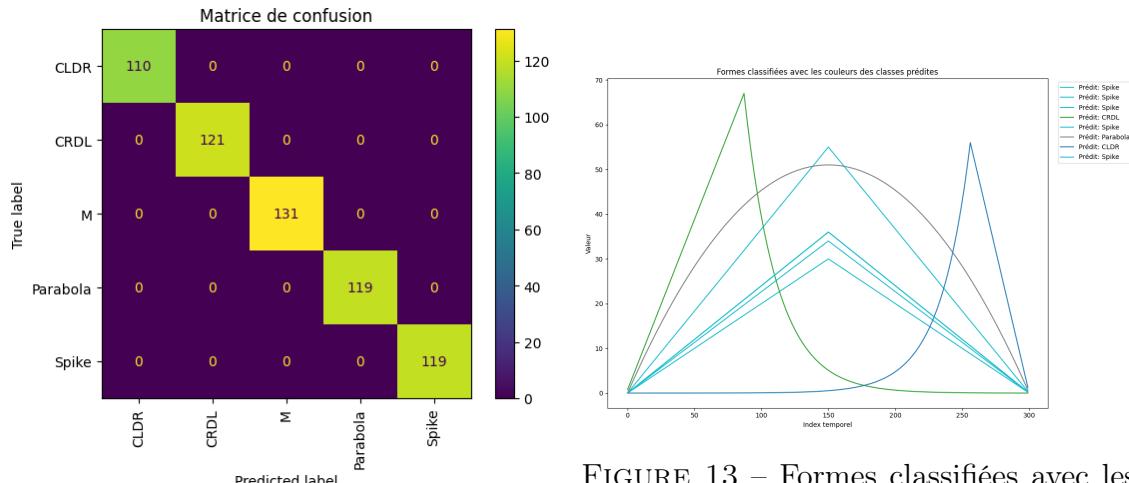


FIGURE 12 – Matrice de confusion des formes complètes et sans bruit.

FIGURE 13 – Formes classifiées avec les couleurs des classes prédictes.

La matrice de confusion (figure 12) montre une classification parfaite, sans aucune confusion entre les classes. Cela reflète la capacité du modèle à exploiter les

caractéristiques des données dans un contexte idéal.

Résultats sur des formes complètes de même largeur avec un bruit faible ($\sigma = 1$)

Après entraînement du **modèle simple** sur des données contenant un bruit modéré ($\sigma = 1$), les résultats obtenus sont les suivants :

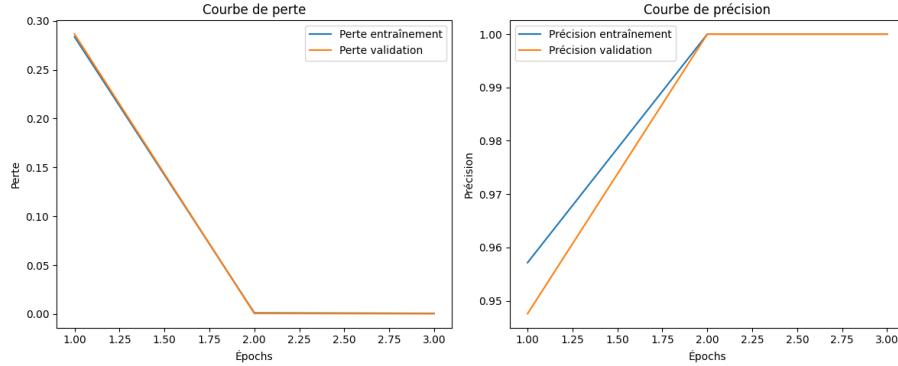


FIGURE 14 – Courbes de perte et de précision pour l’entraînement et la validation avec un bruit $\sigma = 01$.

Comme illustré dans la figure 14, le modèle converge rapidement et atteint une précision quasi-parfaite dès les premières époques. La perte diminue rapidement jusqu’à zéro, ce qui reflète la simplicité du problème malgré l’ajout d’un faible bruit.

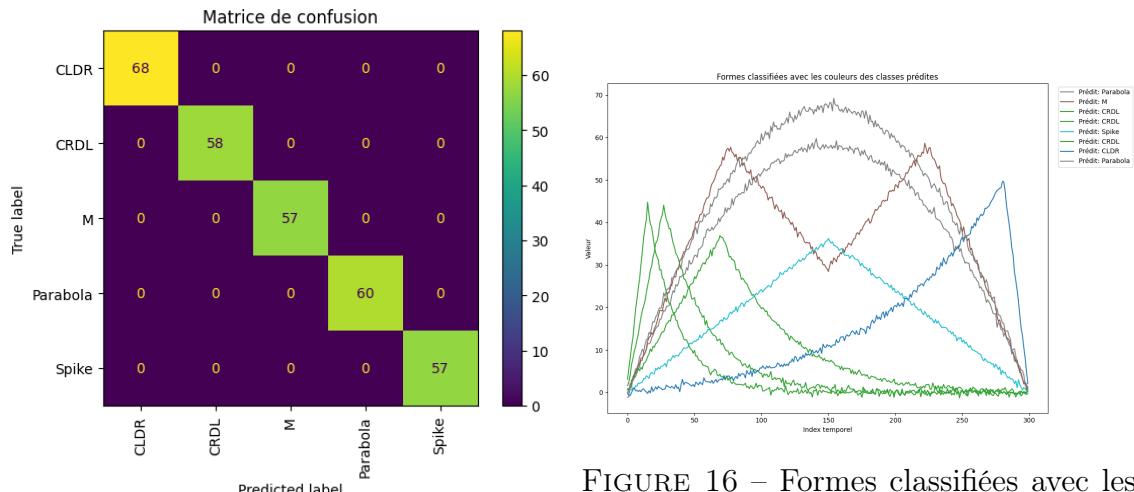


FIGURE 15 – Matrice de confusion des formes avec un bruit $\sigma = 1$.

FIGURE 16 – Formes classifiées avec les couleurs des classes prédictes ($\sigma = 1$).

La matrice de confusion (figure 15) montre une classification parfaite, sans confusion entre les classes, malgré l’ajout de bruit. Cela indique que le modèle est capable

d'extraire les principales caractéristiques des données, même perturbées. Les formes classifiées (figure 16) confirment que le bruit à un niveau faible ($\sigma = 1$) n'altère pas significativement les courbes.

Résultats sur des formes complètes de même largeur avec un bruit réel ($\sigma = 2.22$)

Après entraînement du **Modèle Optimal** sur des données contenant un bruit réaliste ($\sigma = 2.22$), les résultats obtenus sont les suivants :

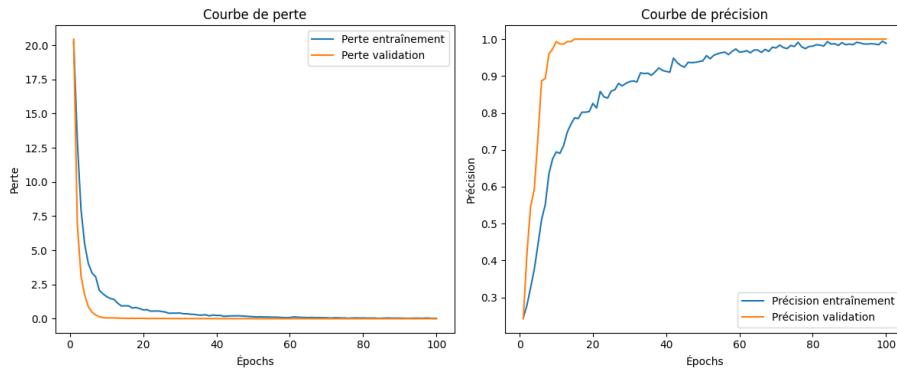


FIGURE 17 – Courbes de perte et de précision pour l'entraînement et la validation avec un bruit $\sigma = 2.22$.

Comme illustré dans la figure 17, le modèle converge de manière stable malgré la présence d'un bruit réaliste.

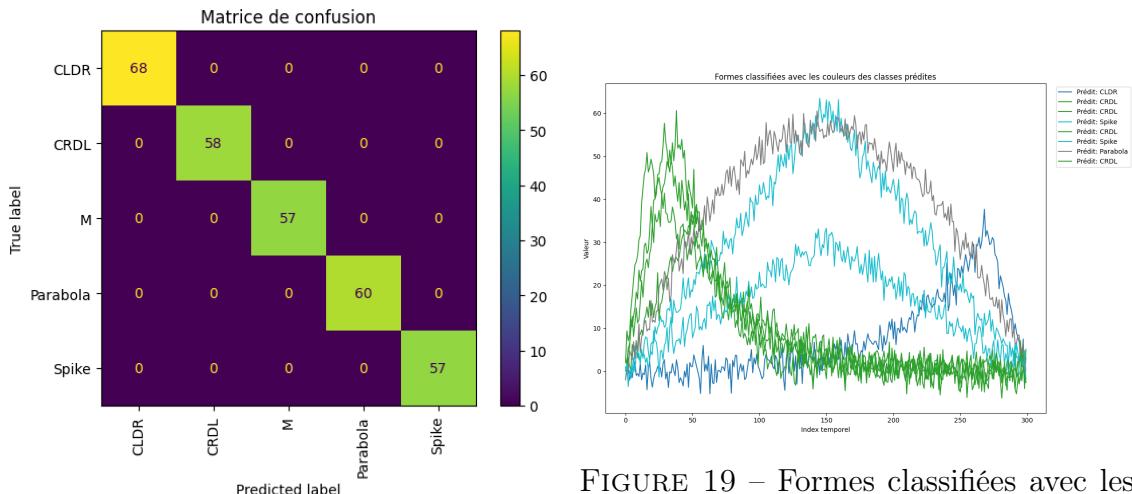


FIGURE 18 – Matrice de confusion des formes avec un bruit $\sigma = 2.22$.

FIGURE 19 – Formes classifiées avec les couleurs des classes prédictes ($\sigma = 2.22$).

La matrice de confusion (figure 18) indique une classification correcte pour les

échantillons, malgré un bruit réaliste. Cela démontre la capacité du modèle à généraliser sur des données réelles. Les formes classifiées (figure 19) confirment que le bruit introduit des variations visibles, mais ces perturbations n'affectent pas significativement la capacité du modèle à identifier correctement les classes.

Résultats sur des formes complètes de même largeur avec un bruit très élevé ($\sigma = 10$)

Changer pour $\sigma = 10$

Après entraînement du **modèle optimal** sur des données fortement bruitées ($\sigma = 1$), les résultats obtenus montrent les limites du modèle face à un bruit important.

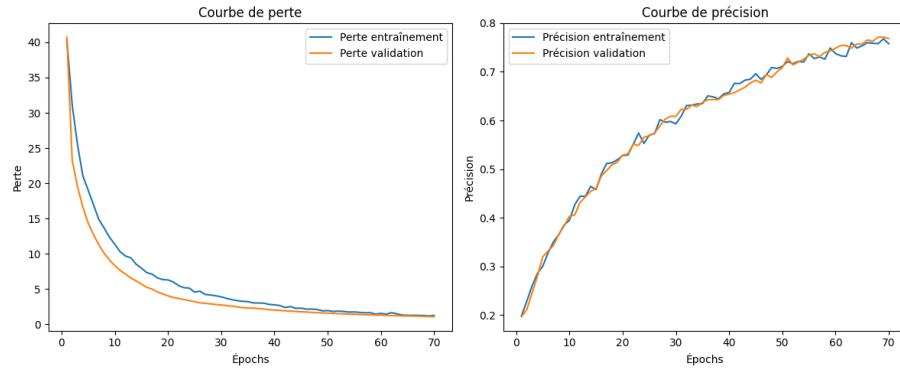
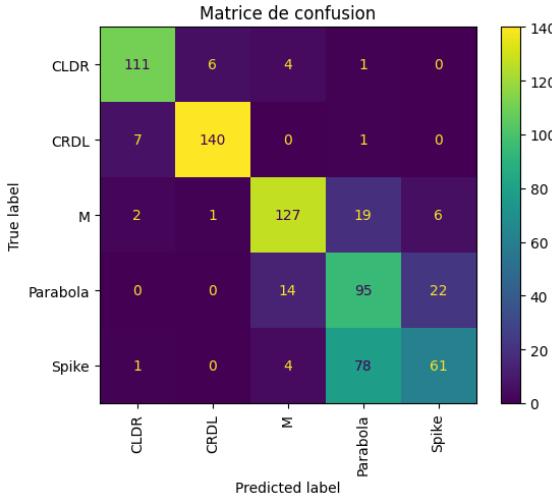


FIGURE 20 – Courbes de perte et de précision pour l'entraînement et la validation avec un bruit $\sigma = 1$.

Comme illustré dans la figure 20, le modèle converge lentement et atteint une précision maximale d'environ 70%. Le bruit élevé complique l'apprentissage des caractéristiques distinctives de chaque classe.


 FIGURE 21 – Matrice de confusion des formes avec un bruit $\sigma = 1$.

La matrice de confusion (figure 21) montre une augmentation significative des confusions entre certaines classes, notamment entre *Parabola*, *Spike*, et *M*. Cela s'explique par le fait qu'avec un bruit très élevé ($\sigma = 1$), les formes de ces classes, qui se concentrent principalement au centre, deviennent difficilement distinguables, entraînant une confusion accrue.

En revanche, les classes *CLDR* et *CRDL* montrent une séparation nette, car leurs caractéristiques distinctives se concentrent sur les hauteurs aux extrémités, rendant leur distinction plus évidente. L'algorithme parvient donc à capter cette séparabilité, ce qui est logique compte tenu de leur positionnement unique dans l'espace des données.

Résultats sur des formes coupées sans bruit

Les formes coupées et de différentes largeurs sans bruit ont été classifiées parfaitement, montrant que le modèle optimal peut gérer efficacement ces données :

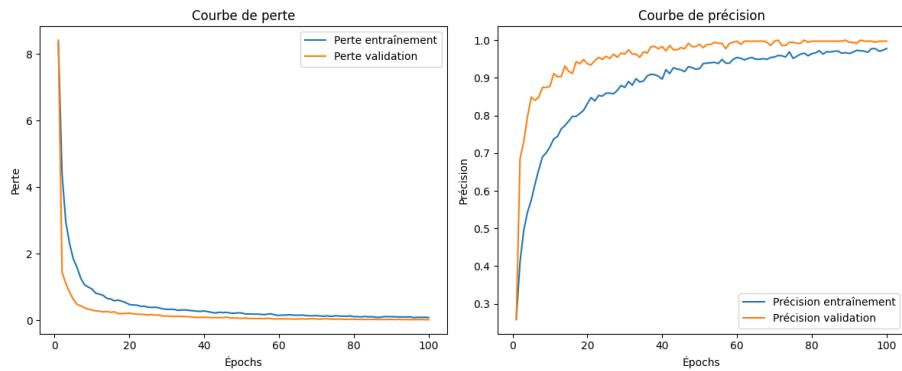


FIGURE 22 – Courbes de perte et de précision pour l'entraînement et la validation des formes coupées sans bruit.

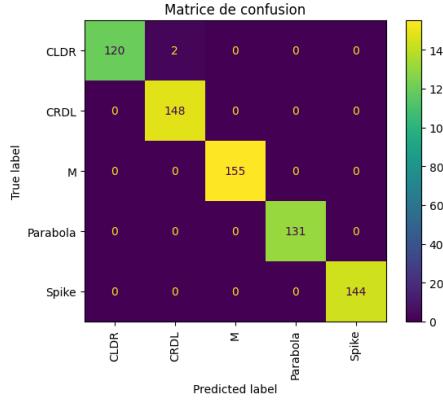


FIGURE 23 – Matrice de confusion des formes coupées sans bruit.

Le modèle a appris rapidement, avec une précision parfaite et une perte minimale.

Résultats sur des formes coupées avec un bruit réel ($\sigma = 2.22$)

Remplacer par $\sigma = 2.22$

Le modèle a montré une bonne robustesse face aux données coupées avec un bruit réel ($\sigma = 0.02$) :

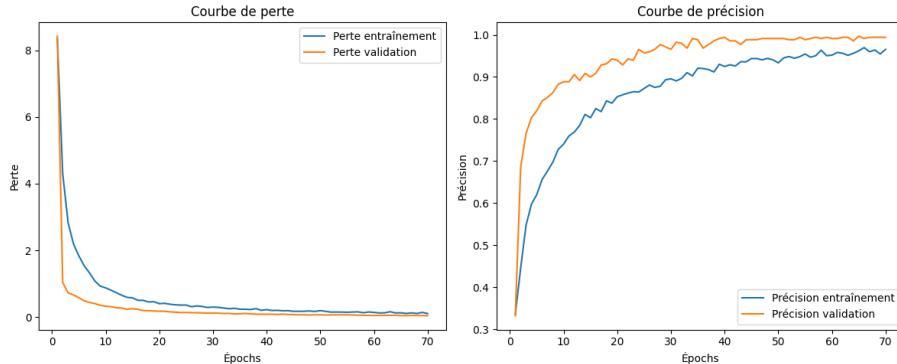
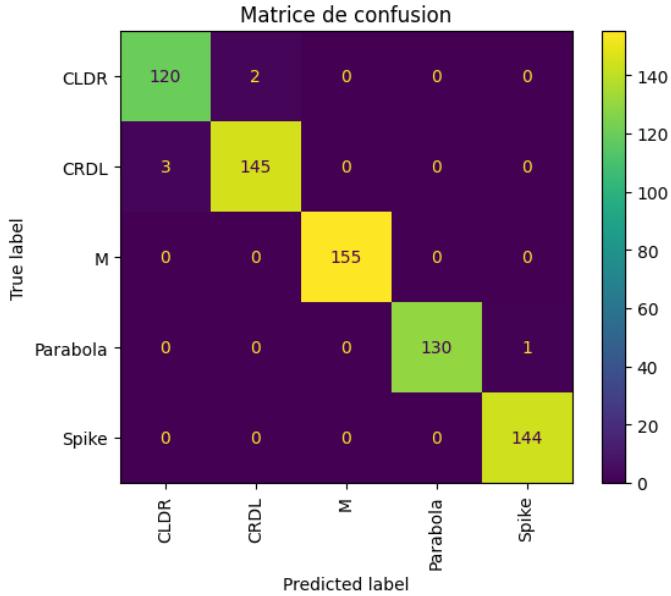


FIGURE 24 – Courbes de perte et de précision pour l'entraînement et la validation des formes coupées avec un bruit $\sigma = 0.02$.


 FIGURE 25 – Matrice de confusion des formes coupées avec un bruit $\sigma = 0.02$.

La matrice de confusion montre que les confusions se limitent aux classes CLDR et CRDL et qu'elles sont très limitées.

Résultats sur des formes coupées avec un bruit mélangé ($\sigma \in \{2.22, 5.0, 10.0\}$)

L'effet d'un bruit mélangé sur des formes coupées et de largeurs différentes a été étudié. Les résultats montrent une bonne capacité du modèle à généraliser malgré la complexité accrue des données.

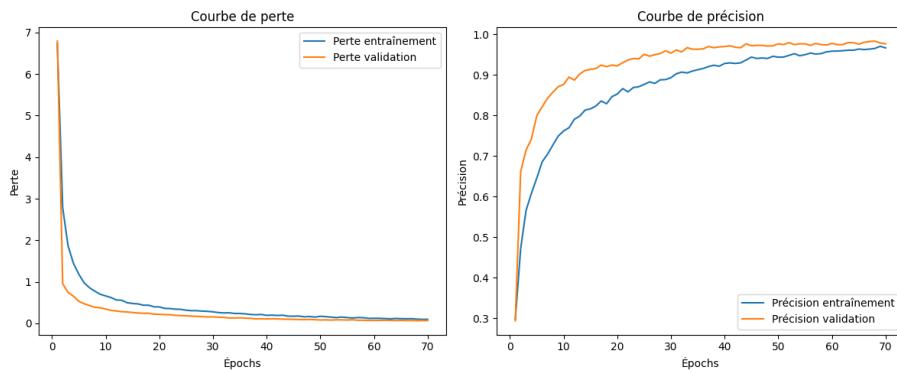


FIGURE 26 – Courbes de perte et de précision pour l'entraînement et la validation des formes coupées avec un bruit mélangé.

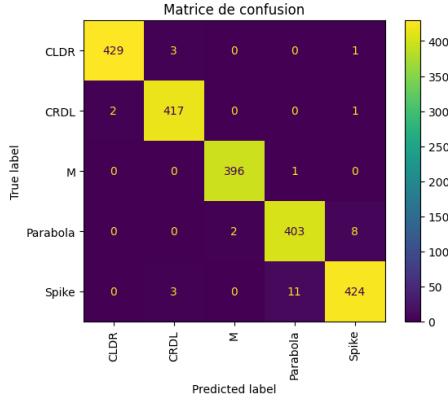


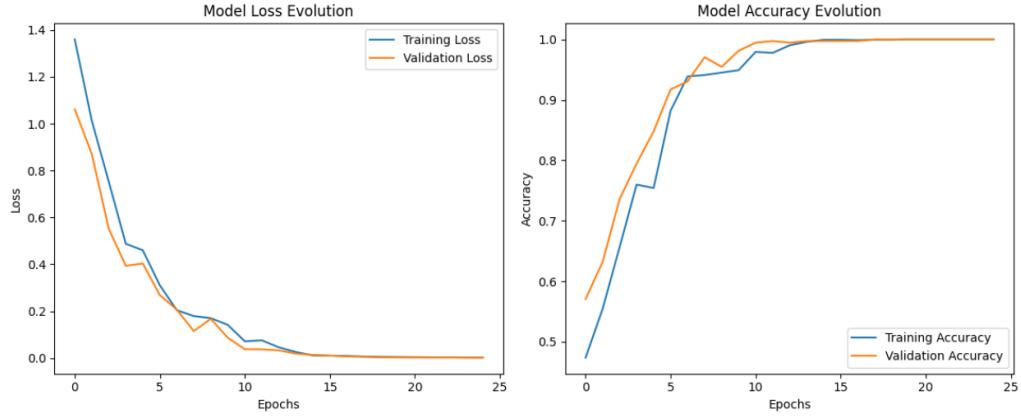
FIGURE 27 – Matrice de confusion des formes coupées avec un bruit mélangé ($\sigma \in \{2.22, 5.0, 10.0\}$).

Les courbes de perte et de précision (figure 26) montrent que le modèle converge correctement malgré la complexité accrue des données. La matrice de confusion (figure 27) indique que les confusions se concentrent principalement entre les classes CLDR et CRDL, ainsi qu'entre Parabola et Spike. Ces confusions sont compréhensibles, car ces classes partagent des caractéristiques similaires qui peuvent être altérées par le bruit et les variations des formes.

4.2 Recurrent Neural Network (LSTM)

Résultats pour des données non bruitées

On entraîne d'abord le modèle sur des données non bruitées. On obtient les précisions et les pertes suivantes :



On obtient une précision de 100% et la matrice de confusion suivante :

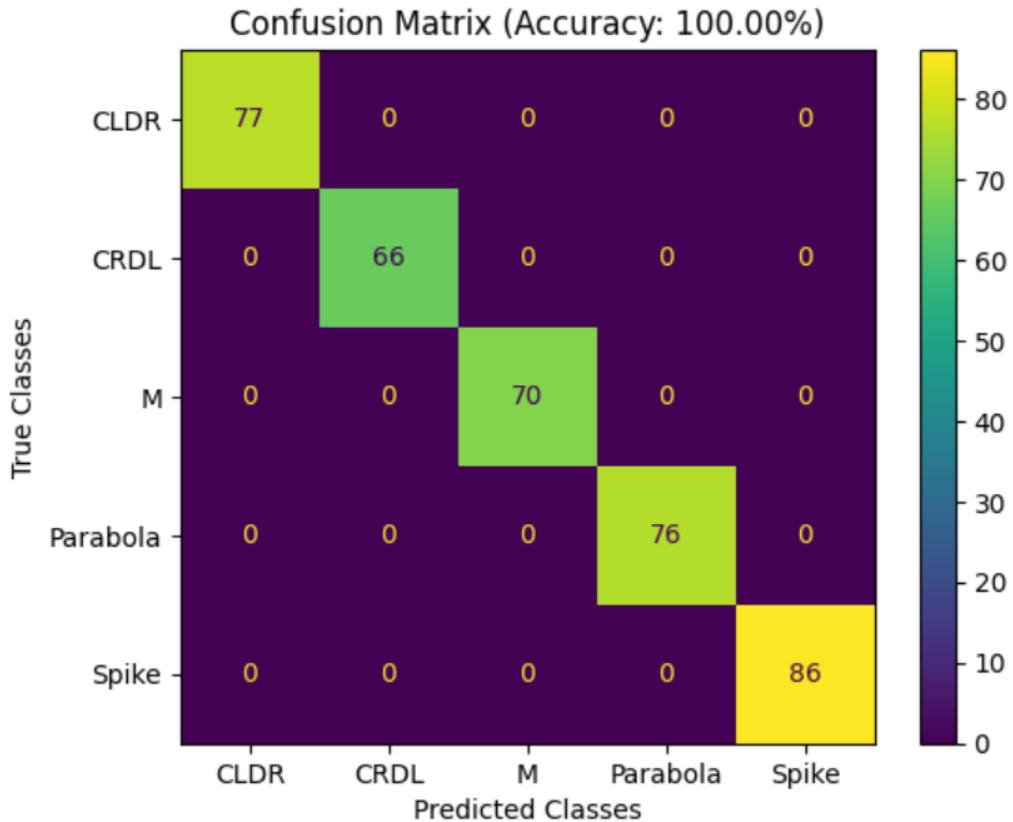


FIGURE 28 – Matrice de confusion pour des données non bruitées

Et enfin on trace les scores de précision, de rappel et la F1 score :

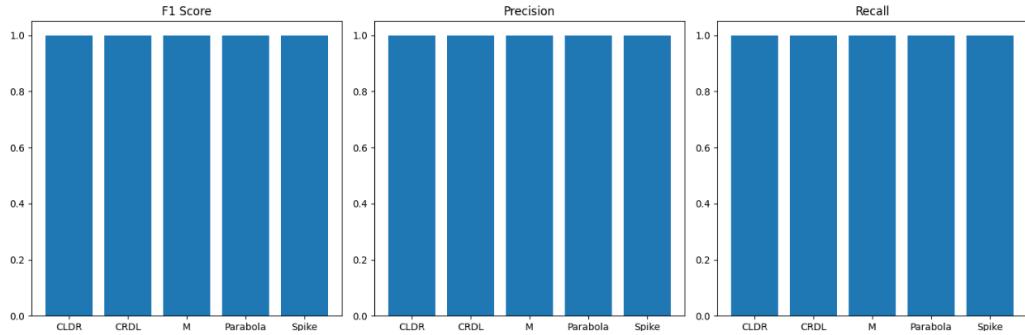
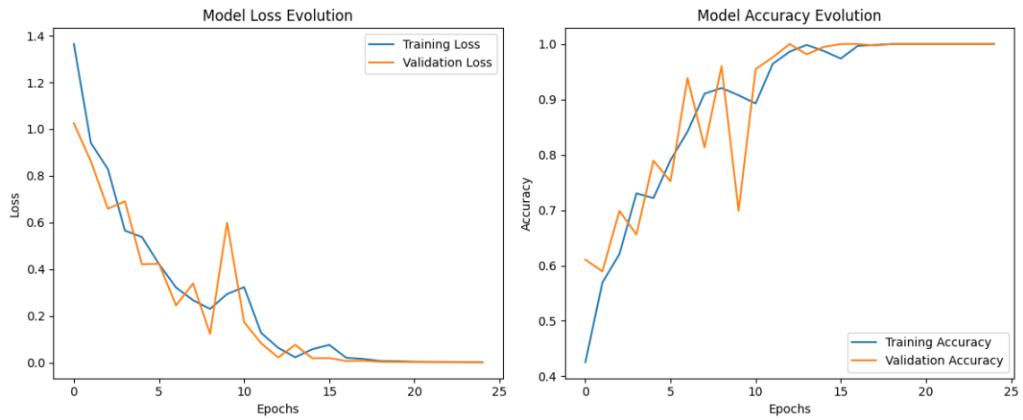


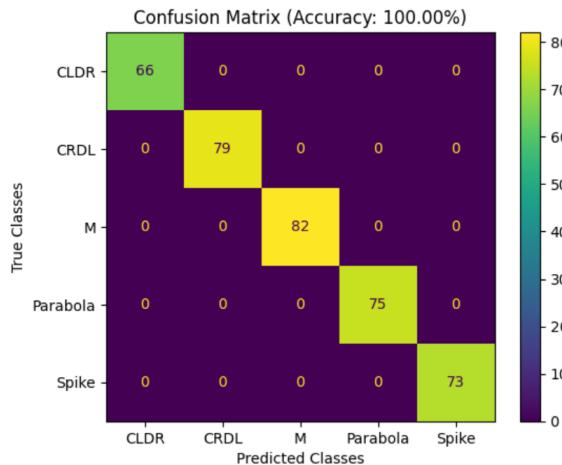
FIGURE 29 – Scores pour des données non bruitées

Résultats pour un bruit d'écart type $\sigma = 2.22$

On entraîne le modèle sur des données faiblement bruitées. On obtient les scores d'accuracy et de pertes suivantes pour le jeu d'entraînement et de validation :


 FIGURE 30 – Accuracy et perte pour un bruit d'écart type $\sigma = 2.22$

On obtient ensuite la matrice de confusion :


 FIGURE 31 – Matrice de confusion pour un bruit d'écart type $\sigma = 2.22$

Et enfin les différents scores :

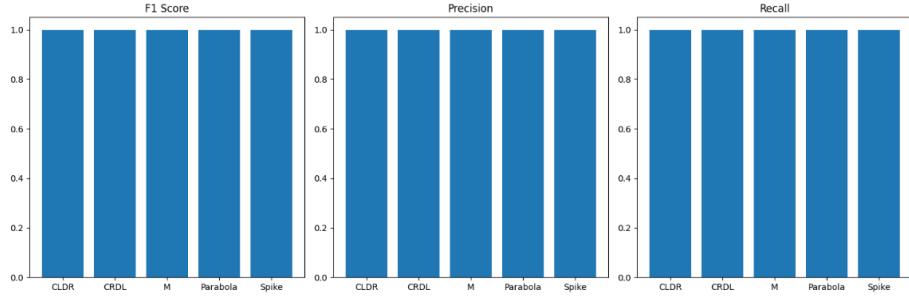


FIGURE 32 – Scores pour un bruit d'écart type de $\sigma = 2.22$

On trace quelques formes prédites :

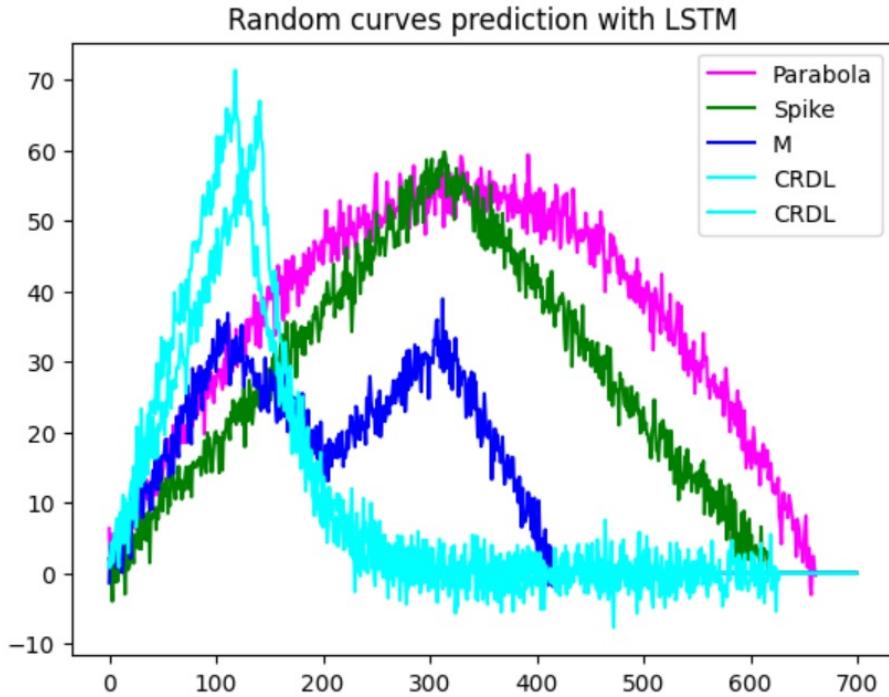


FIGURE 33 – Quelques formes prédites

Résultats pour un bruit d'écart type $\sigma = 5$

On entraîne maintenant le modèle sur des données fortement bruitées. On obtient les scores d'accuracy et de pertes suivantes pour le jeu d'entraînement et de validation :

Classification de motifs d'intérêt en surveillance de l'environnement

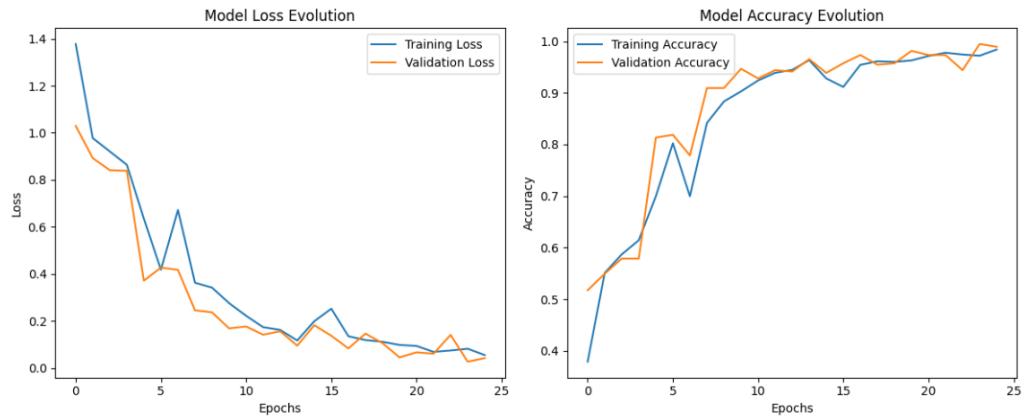


FIGURE 34 – Accuracy et perte pour un bruit d'écart type $\sigma = 5$

On obtient ensuite la matrice de confusion, ainsi que quelques courbes classifiées ;

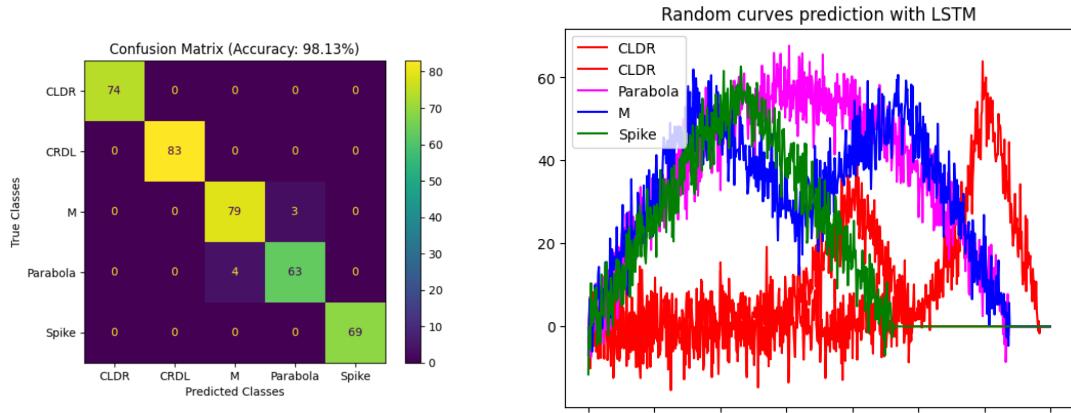


FIGURE 35 – Matrice de confusion pour un bruit d'écart type $\sigma = 5$

FIGURE 36 – Formes classifiées avec les couleurs des classes prédictes ($\sigma = 5$).

Et enfin les différents scores :

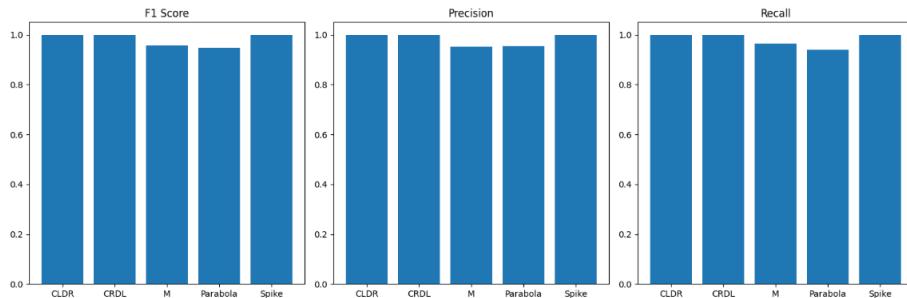


FIGURE 37 – Scores pour un bruit d'écart type de $\sigma = 5$

On remarque que les résultats ne se sont quasiment pas détériorés, ce qui montre la robustesse et l'efficacité du modèle basé sur la cellule LSTM.

Résultats pour des bruits variés

Enfin, pour tester les limites de notre modèle, on le teste pour des bruits différents d'écart types respectifs $\sigma = 0$, $\sigma = 2, 22$, $\sigma = 5$ et $\sigma = 10$.

On obtient les scores d'accuracy et de pertes suivantes pour le jeu d'entraînement et de validation :

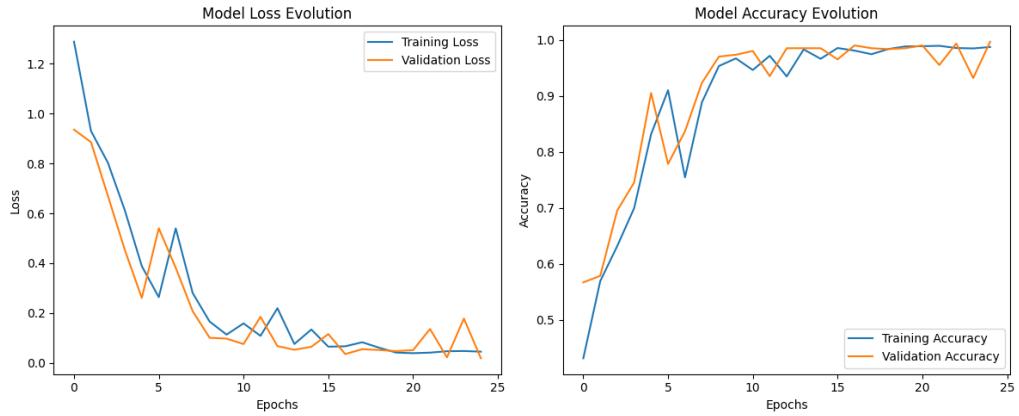


FIGURE 38 – Accuracy et perte pour des bruits variés

On remarque que le modèle met davantage de temps à converger mais les résultats restent satisfaisants. On trace ensuite la matrice de confusion :

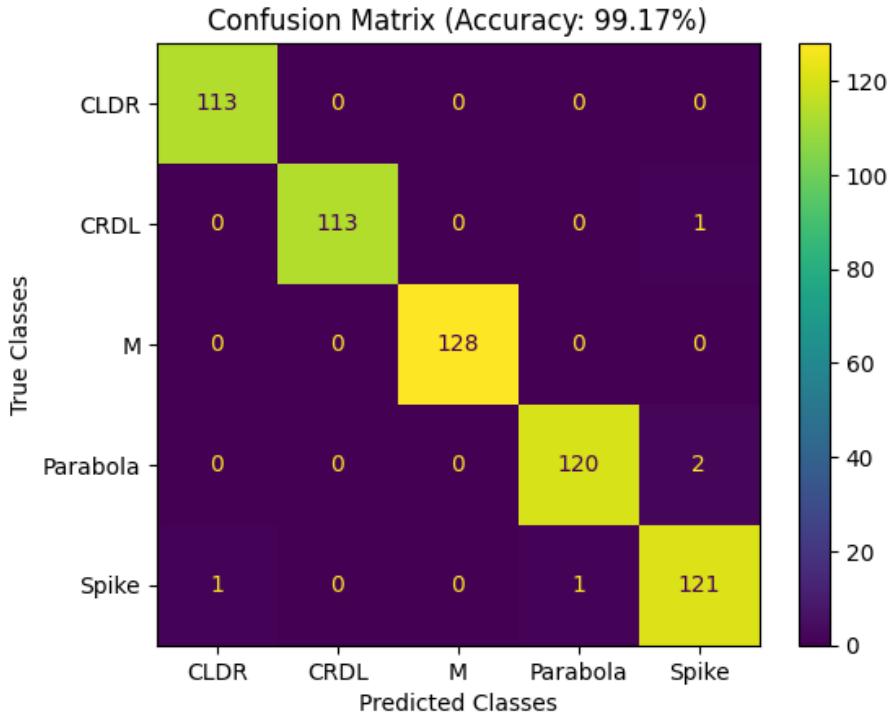


FIGURE 39 – Matrice de confusion pour des bruits variés

Les résultats restent excellents malgré le niveau de bruit important.

Traçons les différents scores :

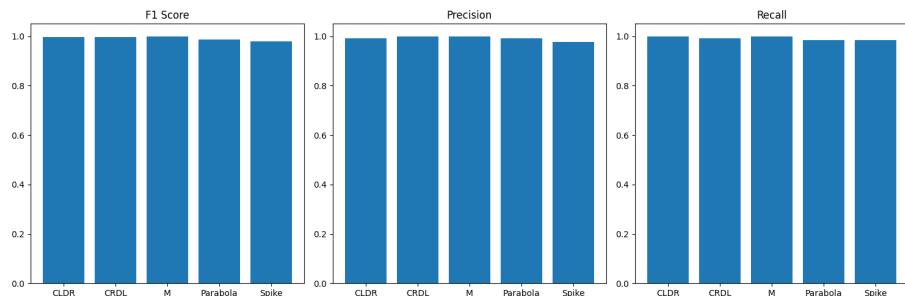


FIGURE 40 – Scores pour des bruits variés

4.3 Comparaisons

On reprend les résultats obtenus dans le rapport de l'année dernière avec U-K-Means [4].

Sans bruit et avec un faible bruit

Pour des données non bruitées, on obtient une précision de 100% pour le modèle ANN et LSTM.

On obtenait l'année dernière une précision de 97% avec l'algorithme U-K-Means. Pour ce qui s'agit des données faiblement bruitées, les modèles basés sur les réseaux de neurones obtiennent des précisions toujours proches de 100%, tandis qu'on avait remarqué une baisse de performances pour l'algorithme U-K-Means avec une précision de 87%

TABLE 2 – Comparaison des performances des modèles avec et sans bruit

Modèle	Accuracy (Sans Bruit)	Accuracy (Faible Bruit)
U-K-Means	97%	87%
ANN	100%	100%
LSTM	100%	100%

Néanmoins, la comparaison avec l'algorithme U-K-Means n'est donnée qu'à titre indicatif, car comparer des méthodes de classification et de clustering non supervisées n'est pas vraiment pertinent.

Avec un bruit plus important

Pour des bruits plus importants ou variés, la comparaison des deux modèles donne :

TABLE 3 – Comparaison des performances des modèles avec et sans bruit

Modèle	Accuracy ($\sigma = 5$)	Accuracy (Bruit varié)
ANN Simple	100%	81%
ANN optimal	100%	93%
LSTM	98,13%	99,17%

On constate que pour des bruits de plus en plus variés et d'écart type de plus en plus élevé, le modèle LSTM donne de meilleurs résultats.

En ce qui concerne les temps d'entraînement, les modèles d'ANN simples s'entraînent plus rapidement que LSTM.

5 Discussions

Les résultats obtenus démontrent une efficacité remarquable des réseaux de neurones, notamment des modèles ANN et LSTM, dans la classification des motifs environnementaux, même en présence de bruit. Les performances, atteignant jusqu'à 100 % de précision dans certains cas, confirment la puissance de ces approches pour capturer des dépendances complexes et gérer des données bruitées.

Cependant, il est important de noter que cette performance élevée s'accompagne de certaines limites et considérations pratiques :

— Comparaison avec U-K-Means

Les modèles basés sur l'apprentissage profond surpassent largement l'algorithme U-K-Means, surtout lorsque les données contiennent du bruit. Toutefois, cette comparaison reste limitée, car U-K-Means est une méthode non supervisée, alors que les réseaux de neurones nécessitent un apprentissage supervisé avec des données étiquetées. Cela rend les résultats dépendants de la qualité et de la représentativité des jeux de données utilisés pour l'entraînement.

— Impact du bruit

Bien que les modèles ANN et LSTM montrent une résilience face à des niveaux de bruit modérés, des scénarios impliquant des bruits variés ou de forte intensité révèlent une légère détérioration des performances, notamment pour les classes les plus ambiguës. Cela souligne l'importance d'une génération de données réalistes et diversifiées pour améliorer la robustesse des modèles.

— Complexité et temps d'entraînement

Les modèles LSTM, bien qu'efficaces, nécessitent des ressources computationnelles importantes et des temps d'entraînement plus longs par rapport aux modèles ANN. Ce coût peut être un frein dans des contextes où les ressources matérielles sont limitées ou lorsque des résultats rapides sont nécessaires.

Pour aller plus loin

Aurélien Géron suggère dans son livre [2] que des couches de convolution 1D peuvent être plus efficaces dans la gestion des séries temporelles, notamment l'architecture WaveNet [5] qui a montré des résultats impressionnantes, notamment pour des séries temporelles plus longues.

Elle se base sur l'empilement de couches de convolution 1D avec taux de dilation croissant comme sur la figure ci-dessous :

Ainsi, les plus hautes couches apprennent les dépendances à long terme tandis que les plus basses couches apprennent les dépendances à court terme.

Nous avons ainsi entraîné le modèle suivant :

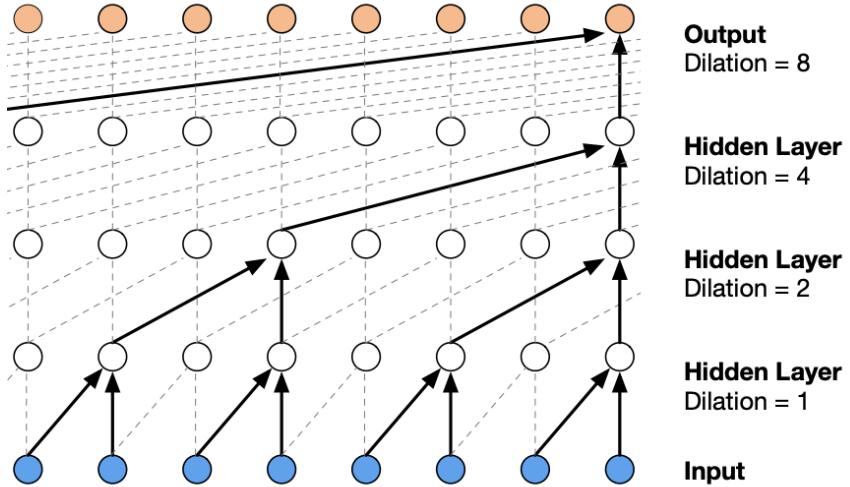


FIGURE 41 – Schéma du modèle WaveNet

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 1197, 64)	320
conv1d_1 (Conv1D)	(None, 1191, 64)	16,448
conv1d_2 (Conv1D)	(None, 1179, 64)	16,448
conv1d_3 (Conv1D)	(None, 1155, 64)	16,448
conv1d_4 (Conv1D)	(None, 1107, 64)	16,448
conv1d_5 (Conv1D)	(None, 1011, 64)	16,448
conv1d_6 (Conv1D)	(None, 1008, 64)	16,448
conv1d_7 (Conv1D)	(None, 1002, 64)	16,448
conv1d_8 (Conv1D)	(None, 990, 64)	16,448
conv1d_9 (Conv1D)	(None, 966, 64)	16,448
conv1d_10 (Conv1D)	(None, 918, 64)	16,448
conv1d_11 (Conv1D)	(None, 822, 64)	16,448
GlobalAveragePooling1D (GlobalAveragePooling1D)	(None, 64)	0
Dropout_Layer_1 (Dropout)	(None, 64)	0
Output_Layer (Dense)	(None, 5)	325

FIGURE 42 – Implémentation de WaveNet

Nous avons testé sur des données comprenant plus de points et avec un bruit varié et important. On obtient alors les résultats impressionnantes suivants.

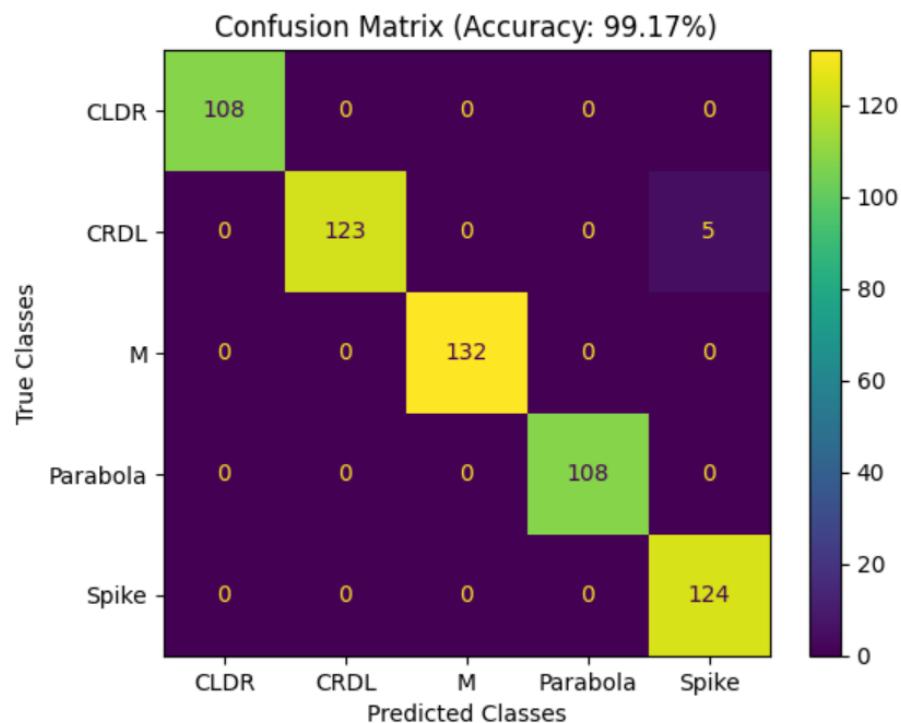


FIGURE 43 – Matrice de confusion pour le modèle WaveNet pour un bruit varié

6 Conclusion

Ce travail a exploré l'utilisation des réseaux de neurones, notamment les modèles ANN et LSTM, pour la classification de motifs environnementaux dans le cadre de la surveillance de l'environnement. Les résultats obtenus mettent en évidence la robustesse et l'efficacité de ces approches, en particulier face à des données bruitées, avec des taux d'accuracy atteignant jusqu'à 100 % dans bien des scénarios. Les tests ont également démontré la résilience des modèles face à des niveaux de bruit modérés, bien que des scénarios plus complexes révèlent des limites dans la capacité de certains modèles à généraliser.

La comparaison avec des méthodes traditionnelles, comme U-K-Means, confirme la supériorité des modèles basés sur l'apprentissage profond en termes de précision et de capacité à traiter des données complexes. Cependant, il convient de noter que ces modèles nécessitent des données étiquetées et des ressources computationnelles significatives, ce qui peut limiter leur adoption dans certains contextes.

Les perspectives ouvertes par cette étude sont prometteuses. À l'avenir, des recherches pourraient explorer l'intégration de mécanismes d'explicabilité pour faciliter l'interprétation des prédictions par les experts, ou encore le développement d'architectures hybrides combinant les avantages des méthodes supervisées et non supervisées. De plus, l'application à des données réelles dans des contextes variés permettrait de valider davantage la robustesse et l'utilité de ces modèles pour des applications concrètes.

En résumé, ce projet représente une avancée notable dans la classification automatique de motifs environnementaux et souligne le potentiel des approches basées sur l'apprentissage profond pour améliorer la précision et l'efficacité des systèmes de surveillance environnementale.

Bibliographie

- [1] Kyle HUNDMAN et al. “Detecting spacecraft anomalies using lstms and non-parametric dynamic thresholding”. In : *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, p. 387-395.
- [2] Aurélien GÉRON. *Deep Learning avec Keras et TensorFlow*. fr. Dunod, 2021. ISBN : 978-2212678136.
- [3] Kristina P SINAGA et Miin-Shen YANG. “Unsupervised K-means clustering algorithm”. In : *IEEE access* 8 (2020), p. 80716-80727.
- [4] ABENHAIM, ALIMI, KEMICHE et ZAAFOURI. “Rapport de projet S6 : Classification de motifs d'intérêt en surveillance de l'environnement”. In : *CS* (2024).
- [5] Aaron van den OORD et al. “WaveNet : A generative model for raw audio”. In : *arXiv preprint arXiv :1609.03499* (2016).
- [6] Saeed AGHABOZORGI, Ali Seyed SHIRKHORSHIDI et Teh Ying WAH. “Time-series clustering—a decade review”. In : *Information systems* 53 (2015), p. 16-38.