

# Détection d'Anomalies dans les Signaux IQ

## Deep Learning for Signal Processing

Yanis Allal & Marine Vieillard

Master MVA et Master Maths&IA

16 décembre 2025

# Plan de la présentation

- 1 Classification des signaux IQ
- 2 Première approche : seuil de confiance
- 3 Analyse des données
- 4 Notre détecteur : approche k-NN multi-échelle
- 5 Conclusion

**Objectif :** Classifier des signaux radio IQ en 6 types de modulation

**Données :**

- Signaux IQ : 2 canaux (I, Q)  $\times$  1024 échantillons
- 6 classes connues (C0-C5)
- SNR variable : 0, 10, 20, 30 dB
- Train : 30000, Val : 10000, Test : 2000 signaux

**Défi :** 3 classes inconnues (C6, C7, C8) apparaissent au test (pas de SNR = 0 dB)

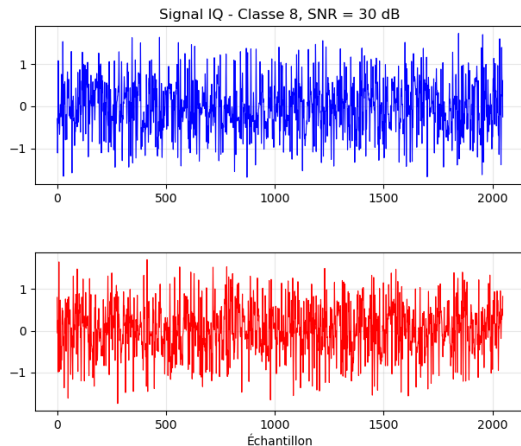
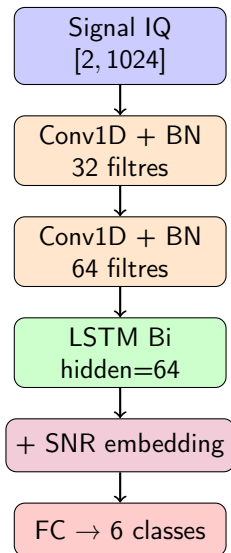


Figure – Exemple de signal

# Architecture du classifieur : CNN-LSTM



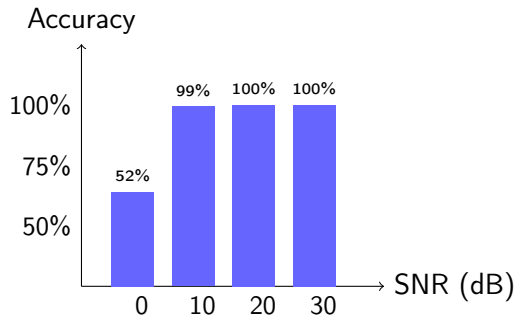
## Pourquoi cette architecture ?

- **CNN** : Extraction des motifs locaux de modulation
- **LSTM bidirectionnel** : Capture des dépendances temporelles
- **SNR embedding** : Le bruit impacte fortement la classification

## Entraînement :

- 500 epochs, batch=512
- Adam + CrossEntropy
- Augmentation : rotation de phase

## Accuracy par SNR



## Résultats :

- Accuracy globale : **88%**
- Excellente performance à  $\text{SNR} \geq 10 \text{ dB}$
- Difficultés à  $\text{SNR} = 0 \text{ dB}$  (signal très bruité)

# Approche naïve : seuil sur la confiance

## Idée intuitive :

Un signal inconnu devrait générer une confiance plus faible

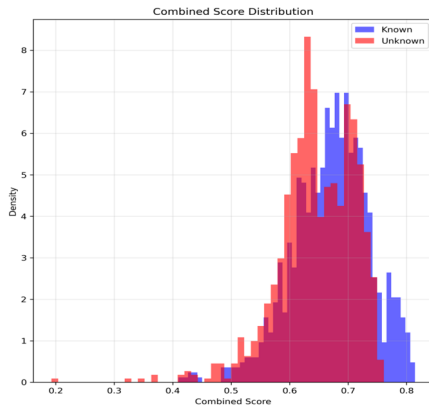
## Méthode :

$$\text{conf} = \max_i \text{softmax}(z)_i$$

$$\text{Décision} = \begin{cases} \text{Inconnu} & \text{si } \text{conf} < \tau \\ \text{Connu} & \text{sinon} \end{cases}$$

## Problème : Classes inconnues

Le classifieur assigne les classes C6, C7, C8 aux classes connues avec **haute confiance**.



**Les distributions se chevauchent !**

# Visualisation t-SNE des embeddings



## Observation clé :

Les classes inconnues ne sont pas séparées.

Elles forment des **clusters distincts** mais **proches** des classes connues.

## Implications

- Distance au centroïde : **ÉCHEC**
- Détection de périphérie : **ÉCHEC**
- $\Rightarrow$  Il faut une approche **locale**

# Pourquoi les inconnus ressemblent aux connus ?

## Analyse des statistiques :

	Amplitude	Phase	Puissance
C3 (connu)	0.42	0.31	0.18
C6 (inconnu)	0.43	0.30	0.19
C5 (connu)	0.38	0.45	0.15
C7 (inconnu)	0.39	0.44	0.16

⇒ Les classes C6, C7, C8 sont des variantes de C3 et C5

## Confusion du classifieur :

- C6 → prédit comme C3 (72%)
- C7 → prédit comme C5 (68%)
- C8 → prédit comme C3 (75%)

### Insight

Les différences sont subtiles mais détectables dans l'espace des features



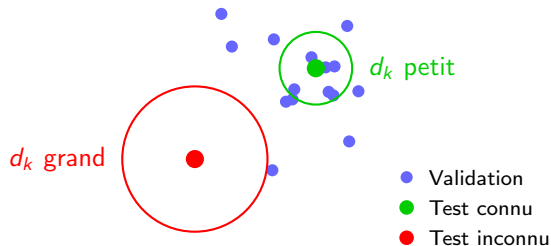
# Principe de l'approche k-NN

## Idée fondamentale :

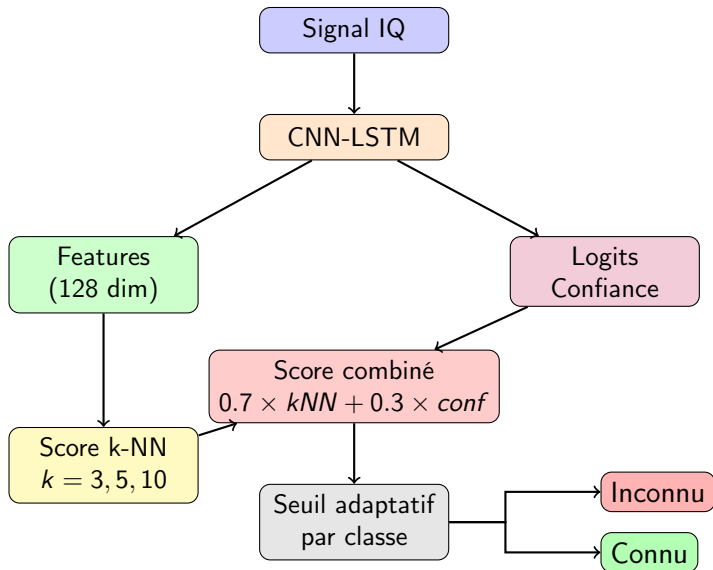
- Signal **connu** → proche de voisins similaires
- Signal **inconnu** → légèrement plus éloigné

## Méthode :

- 1 Extraire les features CNN du signal
- 2 Calculer la distance au  $k$ -ème voisin dans la validation
- 3 Comparer à la distribution des distances connues



# Architecture du détecteur



# Score k-NN multi-échelle

## Étape 1 : Normalisation des features

$$f_{norm} = \frac{f - \mu_{val}}{\sigma_{val}}$$

## Étape 2 : Distance au $k$ -ème voisin

$d_k(x)$  = distance au  $k$ -ème plus proche voisin dans  $\mathcal{D}_{val}$

## Étape 3 : Score percentile

$$\text{score}_k = 1 - \frac{|\{x' \in \mathcal{D}_{val} : d_k(x') > d_k(x)\}|}{|\mathcal{D}_{val}|}$$

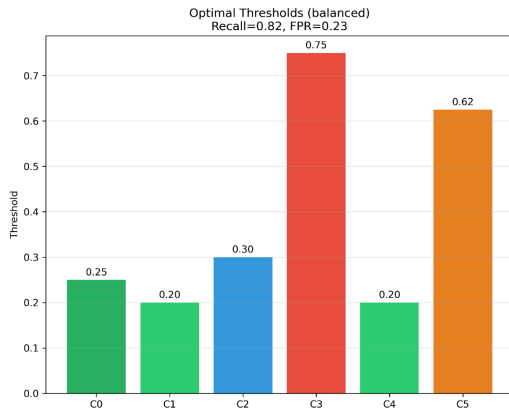
## Étape 4 : Combinaison multi-échelle

$$\text{score}_{kNN} = \underbrace{0.4}_{\text{sensible}} \times \text{score}_{k=3} + 0.35 \times \text{score}_{k=5} + \underbrace{0.25}_{\text{robuste}} \times \text{score}_{k=10}$$

# Seuils adaptatifs par classe

**Observation :** Toutes les classes ne sont pas égales

Classe	Confusion	Seuil
C0	Moyenne	0.25
C1	Faible	0.20
C2	Moyenne	0.30
C3	Élevée	0.75
C4	Faible	0.20
C5	Élevée	0.62

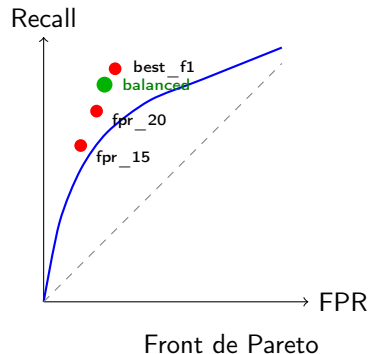


Classes sûres → seuil bas  
Classes risquées → seuil haut

## Points de fonctionnement Pareto :

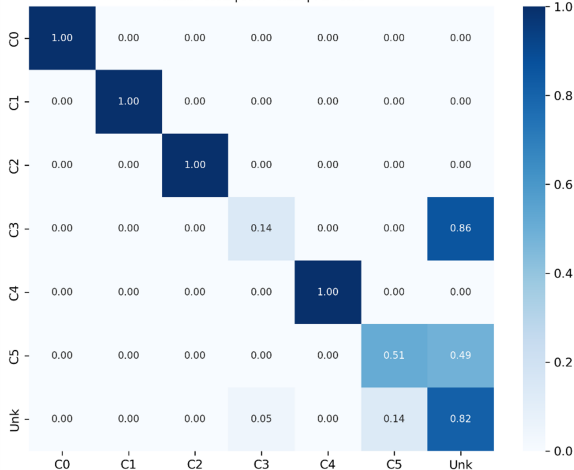
Point	Recall	Precision	FPR	F1
fpr_15	59%	80%	14%	0.68
fpr_20	72%	78%	20%	0.75
<b>balanced</b>	<b>82%</b>	<b>77%</b>	<b>23%</b>	<b>0.79</b>
best_f1	88%	76%	27%	0.82

AUC : 0.74



# Matrice de confusion (point balanced)

balanced  
Recall=0.82 | FPR=0.23 | F1=0.79



## Observations :

- C0, C1, C2, C4 : 100% correct
- C5 : 51% correct, 49% rejeté
- C3 : 14% correct, 86% rejeté
- Inconnus : 82% détectés

## Compromis

C3 et C5 est très proche des inconnus → difficile à distinguer

## Contributions :

- ① Classifieur CNN-LSTM performant (88% accuracy)
- ② Détecteur OOD basé sur k-NN multi-échelle
- ③ Seuils adaptatifs par classe
- ④ Front de Pareto pour différents cas d'usage

## Résultats finaux :

- Recall : **82%** des anomalies détectées
- Précision : **77%**
- FPR : **23%** de faux positifs
- F1-Score : **0.79**

## Perspectives :

- Seuils adaptatifs au SNR

# Merci de votre attention !

**Code disponible sur GitHub :** <https://github.com/MV-13/MVA-Project-Anomaly-detection>