



Détection et prédition des défauts ferroviaires à partir de données multi-capteurs en machine learning

TANNAOUI MOHAMMED AMINE

SCIENCES DES DONNÉES ET INTÉLLIGENCE ARTIFICIELLE

Mars 2025 - Août 2025

Tuteurs école :
BENZINU ABDESSLAM

Tuteur entreprise :
MAYOLLE QUENTIN

Remerciements :

Je tiens à exprimer ma profonde gratitude à toute l'équipe de Railenium pour m'avoir offert l'opportunité de réaliser ce stage au sein de leur organisation.

Je remercie tout particulièrement mon encadrant, Quentin Mayolle, pour ses points réguliers chaque semaine, qui ont permis de suivre l'avancement de mon travail, ainsi que pour ses nombreux conseils et suggestions d'expérimentations.

Je souhaite également remercier Monsieur David Sodoyer pour son soutien constant et ses précieux conseils tout au long de cette période.

Je remercie Madame Nouha Jaoua pour l'aide qu'elle m'a apportée durant le stage.

Enfin, j'adresse mes remerciements à Messieurs Denovan Lampin et Quentin ROBITAILLIE pour leur accompagnement concernant l'utilisation de GitLab et la maîtrise de l'outil Git.

Je remercie chaleureusement toute l'équipe de Railenium pour leur soutien tout au long de mon stage.

Table des matières

1 Introduction	4
1.1 Présentation de l'entreprise	4
1.1.1 Activités	4
1.1.2 Organisation structurelle	4
1.2 Projet TELLIT	5
1.2.1 Description du Projet TELLIT	5
1.2.2 Problème général	6
2 État de l'art sur les méthodes de fusion de données	9
2.1 Fusion Tardive	9
2.1.1 Théorie de Dempster-Shafer	9
2.1.2 Vote majoritaire	10
2.1.3 Méthode Borda Count	10
2.1.4 Fusion par régression logistique	11
2.1.5 Méthode BKS (Behavior Knowledge Space)	12
2.1.6 Méthode Locality-Based (k-NN)	12
2.2 Fusion Précoce	12
2.2.1 Concaténation des caractéristiques brutes	12
2.2.2 Fusion via autoencodeurs spécifiques à chaque modalité	13
3 Présentation du dispositif expérimental et préparation des données	14
3.1 Dispositif d'acquisition des données	14
3.2 Description des données acquises	15
3.2.1 Présentation des fichiers MF4	15
3.3 Synchronisation et annotation des données	17
4 Prétraitement et visualisation des données	20
4.1 Extraction des caractéristiques	20
4.1.1 Transformée de Fourier (FFT)	20
4.1.2 Spectrogramme classique	20
4.1.3 Spectrogramme Mel	21
4.1.4 Coefficients MFCC (Mel-Frequency Cepstral Coefficients)	21
4.1.5 Énergie du signal	22
4.1.6 Enveloppe du signal	22
4.2 Visualisation des données en fonction des labels	23
4.2.1 Analyse des Transformées de Fourier (FFT) en fonction des labels	23
4.2.2 Visualisation des spectrogrammes et de l'énergie	24
4.3 Prétraitement des données pour l'apprentissage automatique	25
4.3.1 Réduction de dimensionnalité par PCA	25
4.3.2 Gestion du déséquilibre des classes	25

4.3.3	Création d'échantillons contextuels	25
4.3.4	Construction de séquences pour modèles récurrents	25
4.3.5	Normalisation des données	25
5	Approche d'apprentissage automatique pour la détection des joints	26
5.1	Modélisation basée sur les signaux audio	26
5.1.1	Description des métriques d'évaluation	26
5.1.2	Essai avec des modèles de machine learning classiques	26
5.1.3	Optimisation des hyperparamètres avec Optuna	26
5.1.4	Prétraitements et paramètres des données	28
5.1.5	Optimisation d'XGBoost avec Optuna	28
5.1.6	Entraînement avec des réseaux de neurones récurrents (RNN)	31
5.1.7	Entraînement avec des réseaux de neurones récurrents convolutionnels (C-RNN)	36
5.2	Modélisation basée sur les signaux accéléromètres	47
5.2.1	Accéléromètres utilisés et annotations	48
5.2.2	Extraction des caractéristiques	48
5.2.3	Présentation de l'architecture utilisée	49
5.2.4	Résultats par accéléromètre individuel	50
5.2.5	Fusion Précoce	54
5.2.6	Fusion tardive des accéléromètres	57
5.3	Fusion des modalités audio et accéléromètres	59
5.3.1	Présentation des caractéristiques utilisées	59
5.3.2	Présentation de l'architecture utilisée	60
5.3.3	Résultats des fusions multimodales audio–accéléromètres	61
6	Conclusion	64

1 Introduction

1.1 Présentation de l'entreprise

1.1.1 Activités

L’Institut de Recherche Technologique (IRT) Railenium mène des projets collaboratifs de recherche et d’innovation réunissant le monde universitaire, l’industrie et les autorités publiques. Ces initiatives visent à concevoir, valider et tester des technologies de pointe ainsi que des solutions innovantes pour répondre aux défis sociétaux et technologiques majeurs de l’industrie ferroviaire, tant en France qu’à l’international.

Créé en 2012 dans le cadre du Plan d’Investissements d’Avenir, Railenium est un Institut de Recherche Technologique spécialisé dans les systèmes ferroviaires. Fort de l’expertise de ses équipes, Railenium apporte des compétences clés pour surmonter les barrières technologiques et scientifiques. Implanté dans les Hauts-de-France et en Île-de-France, il participe au financement, l’ingénierie et la gestion de projets collaboratifs, ayant contribué à plus de 80 projets avec plus d’une centaine de partenaires.



FIGURE 2 – Chiffre-clés

1.1.2 Organisation structurelle

Railenium est dirigé par une Direction Générale à laquelle sont rattachés le Responsable de la Communication, le Responsable QHSE (Qualité, Hygiène, Sécurité, Environnement) et le Responsable du Système d’Information.

L’institut est organisé en cinq directions principales : Scientifique, Projets et Expertise, Développement, Ressources Humaines, ainsi que Finance et Juridique.

La Direction Scientifique supervise les projets de recherche fondamentale et appliquée, ainsi que les partenariats avec les universités et les centres de recherche.

La Direction Développement se consacre à l’innovation technologique et à la création de nouveaux produits et services pour le secteur ferroviaire.

La Direction Ressources Humaines gère le recrutement, la formation et le développement des carrières du personnel.

La Direction Finance et Juridique est responsable de la gestion financière de l’institut ainsi que du suivi des aspects juridiques et contractuels des projets.

Enfin, **la Direction Projets et Expertises (DPE)**, dirigée par Arnaud DE-LARUE, est en charge de la gestion des projets de recherche et de l’expertise tech-

nique. La DPE est organisée en trois domaines d'expertise, chacun supervisé par un Responsable de Domaine et assisté par un Référent d'Expertise :

1. Intelligence artificielle, science des données et optimisation.
2. Comportement humain et sécurité des systèmes.
3. Prédiction et gestion de l'énergie et de la durée de vie des actifs énergétiques.

J'ai intégré cette direction, plus précisément le pôle d'expertise en intelligence artificielle, où je travaille dans le domaine de l'aide à la décision, sous la supervision de mon manager, Monsieur Quentin MAYOLLE.

L'organigramme de la structure est illustré ci-dessous :

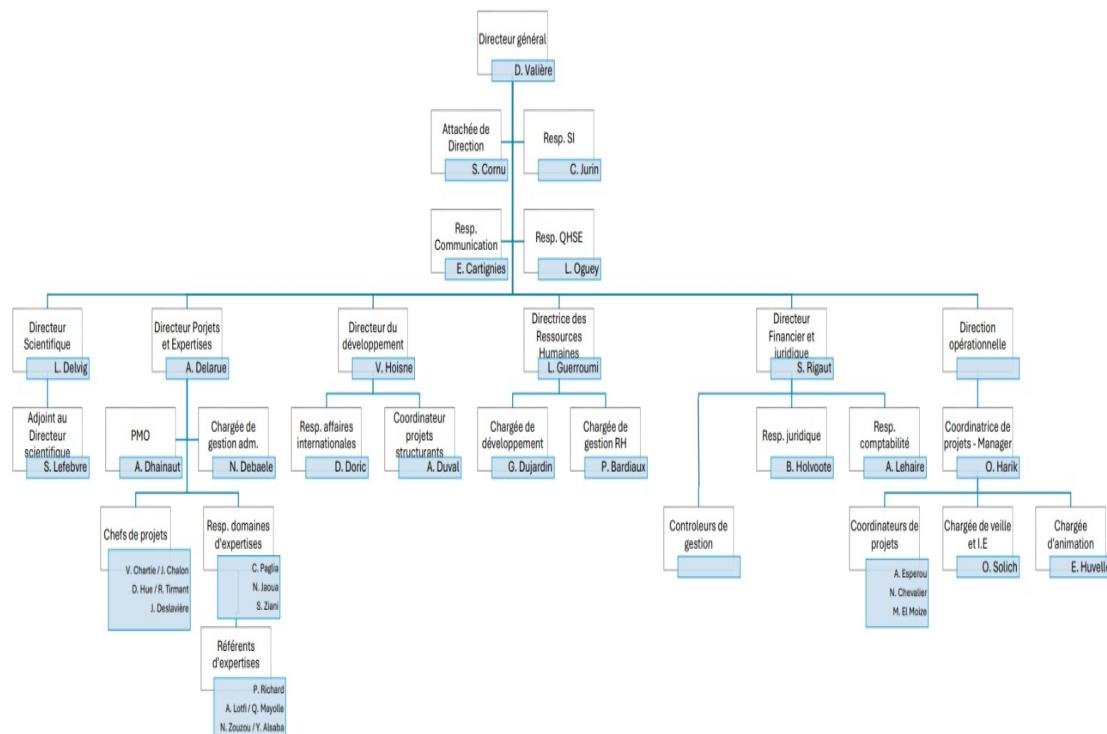


FIGURE 3 – L'organigramme

1.2 Projet TELLi

1.2.1 Description du Projet TELLi

Le projet TELLi est une initiative innovante visant à revitaliser les lignes de desserte fine du territoire français, qui représentent un tiers du réseau ferré national. Conçu pour remplacer les trains diesel par des trains hybrides électriques, TELLi

utilise des sources d'énergie telles que les batteries et l'hydrogène, offrant ainsi une solution de transport plus écologique et économique.

Ce projet se distingue par sa capacité à adapter le matériel roulant aux besoins spécifiques des régions, avec différentes versions selon la source d'énergie, incluant des configurations en courant continu 1500 V, en courant alternatif 25 kV, ainsi que des variantes hydrogène et thermique.

Alimenté par des batteries et fabriqué avec des matériaux recyclables, TELLi est conçu pour être léger, ce qui permet de minimiser la maintenance des voies, tout en offrant une modularité intérieure adaptée à diverses utilisations. Le prototype du Train Léger Innovant (TELLi), illustre cette flexibilité ainsi que l'engagement du projet envers une mobilité durable.

Actuellement, le projet avance avec le soutien de la Région Nouvelle-Aquitaine et de partenaires tels que le Cerema, Alstom, SNCF Réseau, SNCF Voyageur, avec un objectif de commercialisation fixé pour 2029, après une série de démonstrations et de tests prévus sur une ligne laboratoire d'ici 2025.

1.2.2 Problème général

La sécurité et la fiabilité du réseau ferroviaire dépendent en grande partie de l'état des rails. Divers défauts peuvent apparaître au fil du temps, affectant la durabilité du rail et augmentant le risque d'accidents. Parmi ces défauts, on distingue notamment :

- **Squats** : fissures superficielles ou en profondeur apparaissant sur la surface du rail, susceptibles d'évoluer vers des fractures plus graves (voir Fig. 4)[1].
- **Headchecks** : micro-fissures en surface provoquées par le contact répétitif des roues, pouvant entraîner des éclats et des dommages structurels importants du rail (voir Fig. 5)[1].
- **Fissures de rail** : fissures longitudinales ou transversales pouvant fragiliser la structure du rail.
- **Éclats de tête** : éclats ou morceaux manquants sur la tête du rail, souvent causés par le passage intensif des trains.
- **Déformations plastiques** : déformations permanentes du rail (bosses, ondulations) dues à des contraintes mécaniques répétées.
- **Usure latérale** : abrasion excessive sur les flancs du rail, provoquant un profil irrégulier et un mauvais contact roue-rail.
- **Rugosité de surface** : irrégularités fines sur la surface du rail pouvant générer des vibrations et un bruit accru lors du passage des trains.

La détection précoce de ces défauts est cruciale pour planifier la maintenance, éviter les accidents et prolonger la durée de vie des rails. Dans ce contexte, l'analyse de séries temporelles issues de capteurs installés sur les rails et les trains permet d'automatiser la détection des anomalies et d'améliorer la sécurité ferroviaire.



FIGURE 4 – Exemple de squat sur un rail.

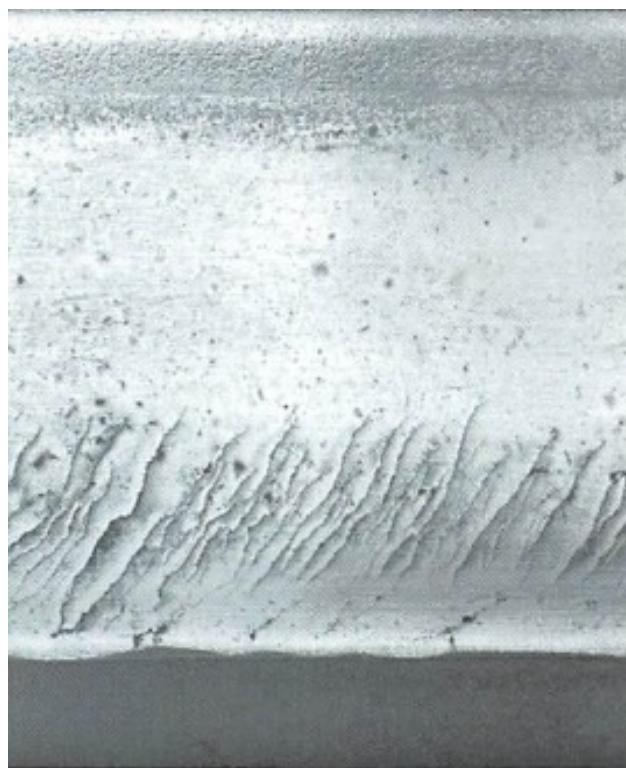


FIGURE 5 – Exemple de headcheck sur un rail.

Le stage de 6 mois chez Railenium en Machine Learning a pour objectifs principaux :

- Réaliser un état de l'art des méthodes de fusion de données.
- Analyser et visualiser les données de séries temporelles.
- Synchroniser et annoter précisément les données temporelles.
- Développer et appliquer des modèles de machine learning sur des séries temporelles.
- Évaluer la performance des modèles et proposer des améliorations adaptées au contexte ferroviaire.

2 État de l'art sur les méthodes de fusion de données

Dans les systèmes de traitement multi-capteurs, la fusion de données est une étape essentielle visant à combiner des informations issues de plusieurs modalités (par exemple : signaux audio, vibrations, images) afin d'améliorer la robustesse, la précision et la cohérence des modèles de détection. On distingue principalement deux approches de fusion : **la fusion précoce**, qui combine les données à un stade initial du pipeline,

$$\mathbf{x}_{fusion} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \quad \Rightarrow \quad \hat{y} = f(\mathbf{x}_{fusion}) \quad (1)$$

et **la fusion tardive**, qui intègre les sorties de modèles indépendants après un traitement séparé de chaque modalité.

$$\hat{y} = g(f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \dots, f_n(\mathbf{x}_n)) \quad (2)$$

où :

- \mathbf{x}_i : caractéristiques extraites du capteur i ,
- \mathbf{x}_{fusion} : vecteur de caractéristiques obtenu par concaténation,
- f : le modèle d'apprentissage automatique utilisé,
- \hat{y} : la prédiction produite par le modèle.

2.1 Fusion Tardive

2.1.1 Théorie de Dempster-Shafer

La théorie de Dempster-Shafer [4] est un cadre mathématique permettant de représenter l'incertitude à l'aide de fonctions de masse. Elle repose sur la notion de *fonction de masse de croyance* m , définie sur l'ensemble des sous-ensembles d'un espace de discernement Θ .

$$m : 2^\Theta \rightarrow [0, 1], \quad \sum_{A \subseteq \Theta} m(A) = 1, \quad m(\emptyset) = 0 \quad (3)$$

Chaque masse $m(A)$ représente la croyance attribuée à une hypothèse A , sans pour autant nécessiter de la distribuer complètement entre les hypothèses individuelles.

La combinaison de deux fonctions de masse m_1 et m_2 , issues de deux sources d'information indépendantes (par exemple, deux capteurs), est réalisée à l'aide de la règle de Dempster :

$$m(A) = \frac{1}{1 - K} \sum_{B \cap C = A} m_1(B) \cdot m_2(C) \quad (4)$$

où K est le facteur de conflit :

$$K = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C) \quad (5)$$

Nous avons choisi d'interpréter ces probabilités comme des fonctions de masse, en attribuant :

- une masse $m_i(\text{défaut}) = \alpha * p_i$
- une masse $m_i(\text{pas de défaut}) = \alpha * (1 - p_i)$
- une masse $m_i(\text{Incertitude}) = 1 - \alpha$

2.1.2 Vote majoritaire

La méthode du vote majoritaire [3] est une technique simple et intuitive de fusion tardive. Elle consiste à agréger les décisions individuelles de plusieurs modèles (ou capteurs) en choisissant la classe prédite par la majorité.

Chaque modèle f_i effectue une prédiction binaire sur la présence (1) ou l'absence (0) d'un défaut à partir de sa propre modalité \mathbf{x}_i :

$$y_i = f_i(\mathbf{x}_i) \in \{0, 1\} \quad (6)$$

La décision finale \hat{y} est obtenue par un vote :

$$\hat{y} = \begin{cases} 1 & \text{si } \sum_{i=1}^n y_i > \frac{n}{2} \\ 0 & \text{sinon} \end{cases} \quad (7)$$

2.1.3 Méthode Borda Count

La méthode Borda [7] Count est une technique de vote préférentiel qui permet de fusionner les prédictions en prenant en compte le classement des hypothèses par chaque modèle.

Chaque modèle f_i fournit un vecteur de probabilités

$$\mathbf{s}_i = [s_{i1}, s_{i2}, \dots, s_{iC}],$$

où s_{ic} représente la probabilité attribuée par le modèle i à la classe c , avec la contrainte suivante :

$$\sum_{c=1}^C s_{ic} = 1 \quad \text{et} \quad s_{ic} \geq 0 \quad \forall c.$$

Pour chaque modèle, les classes sont ordonnées selon leurs scores, et la classe la mieux notée reçoit un score maximal, la seconde un score inférieur, et ainsi de suite.

La contribution Borda $b_i(c)$ de la classe c selon le modèle i est définie comme :

$$b_i(c) = C - \text{rang}_i(c) \quad (8)$$

où $\text{rang}_i(c)$ est la position de la classe c dans le classement des scores du modèle i (1 pour la meilleure, C pour la moins bonne).

La fusion finale des scores Borda pour la classe c est obtenue par sommation :

$$B(c) = \sum_{i=1}^n b_i(c) \quad (9)$$

La classe prédictive \hat{y} correspond alors à celle ayant le score Borda maximal :

$$\hat{y} = \arg \max_{c \in \{1, \dots, C\}} B(c) \quad (10)$$

2.1.4 Fusion par régression logistique

La fusion par régression logistique [9] consiste à utiliser les probabilités issues des modèles individuels comme variables d'entrée d'un modèle de classification supervisée.

Soient p_i les probabilités de présence du défaut fournies par les n modèles pour un échantillon donné, alors la régression logistique estime la probabilité finale \hat{y} de la classe « défaut » par :

$$\hat{y} = \sigma \left(\beta_0 + \sum_{i=1}^n \beta_i p_i \right) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^n \beta_i p_i)}} \quad (11)$$

où σ est la fonction sigmoïde, et les coefficients β_i sont appris lors de l'entraînement du modèle sur un jeu de données annotées.

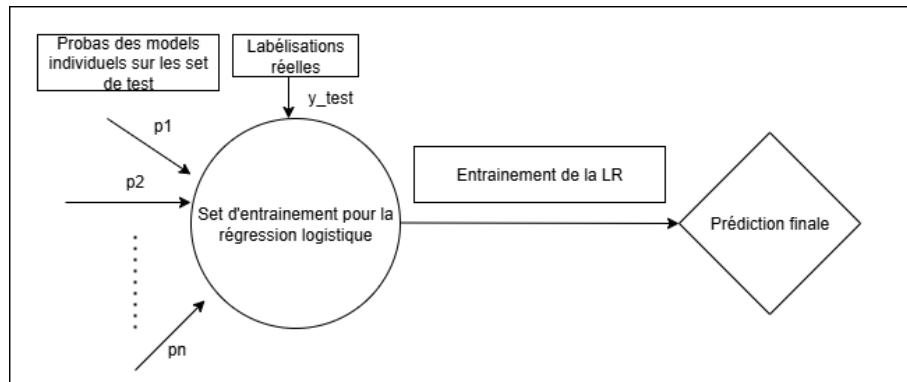


FIGURE 6 – Illustration de la fusion tardive par régression logistique à partir des probabilités issues de modèles individuels

2.1.5 Méthode BKS (Behavior Knowledge Space)

La méthode BKS [7] repose sur la construction d'une table de correspondance entre les sorties des classificateurs individuels et la sortie correcte attendue. Lors de l'entraînement, pour chaque combinaison spécifique de sorties (probabilités, classes, etc.), on enregistre l'étiquette réelle majoritaire associée à cette combinaison. Cette table permet ensuite, lors de l'inférence, de retrouver l'étiquette la plus probable pour une nouvelle combinaison similaire.

Dans notre cas, les clés de la table BKS sont formées à partir des probabilités de sortie des modèles individuels. Pour limiter la complexité de l'espace de recherche, nous avons discrétisé ces probabilités en conservant un seul chiffre après la virgule. Cela permet de réduire le nombre de combinaisons possibles tout en maintenant une granularité raisonnable pour capturer des tendances dans les décisions combinées.

2.1.6 Méthode Locality-Based (k-NN)

La méthode de fusion basée sur la localité [6] consiste à estimer la prédiction finale en s'appuyant sur les décisions d'observations similaires déjà rencontrées. Elle repose généralement sur l'algorithme des k plus proches voisins (k-NN), appliqué dans l'espace des sorties des modèles individuels.

Chaque échantillon de test est comparé aux échantillons de l'ensemble d'entraînement à l'aide d'une mesure de distance (par exemple, la distance euclidienne) calculée entre les vecteurs de probabilités de sortie des modèles. La prédiction finale est ensuite obtenue par vote majoritaire ou moyenne pondérée des labels correspondants aux k voisins les plus proches.

2.2 Fusion Précoce

La fusion précoce [8] consiste à combiner les données issues de différentes modalités avant tout traitement par un modèle d'apprentissage. Cette approche permet de capturer les interactions croisées entre les différentes sources dès l'entrée du réseau.

2.2.1 Concaténation des caractéristiques brutes

La méthode la plus directe consiste à concaténer les vecteurs de caractéristiques extraits de chaque capteur pour former un vecteur global, utilisé comme entrée d'un réseau de classification.(voir Fig. 7).

$$\mathbf{x}_{fusion} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \Rightarrow \hat{y} = f(\mathbf{x}_{fusion}) \quad (12)$$

Cette approche suppose que les vecteurs sont compatibles en termes de taille et de nature, et qu'une simple jointure suffit à exploiter les complémentarités entre les modalités.

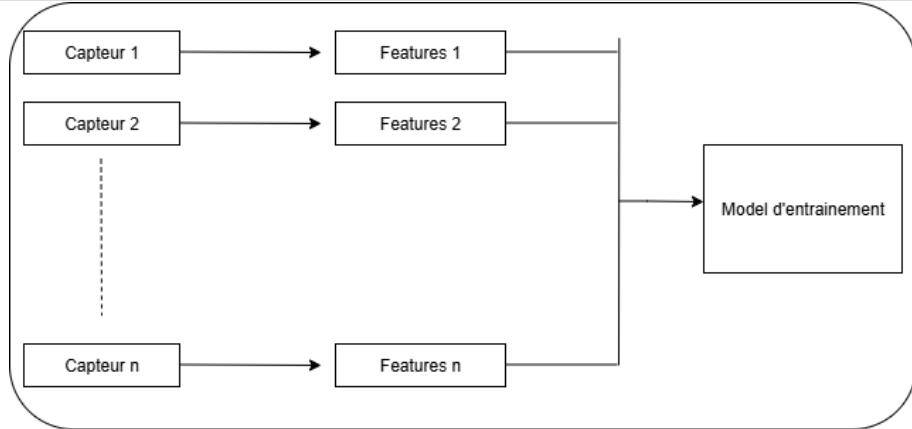


FIGURE 7 – Fusion précoce par concaténation directe des caractéristiques

2.2.2 Fusion via autoencodeurs spécifiques à chaque modalité

Une autre approche consiste à utiliser des autoencodeurs [5] pour chaque capteur afin d'apprendre des représentations compactes (ou embeddings) de chaque modalité. Les représentations encodées sont ensuite fusionnées pour être traitées par un réseau commun.

$$\mathbf{z}_i = \text{Encoder}_i(\mathbf{x}_i), \quad \mathbf{z}_{fusion} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n], \quad \hat{y} = f(\mathbf{z}_{fusion}) \quad (13)$$

Cette stratégie permet de prendre en compte les spécificités de chaque modalité avant de les combiner, tout en réduisant la dimensionnalité globale.

3 Présentation du dispositif expérimental et préparation des données

3.1 Dispositif d'acquisition des données

Les données utilisées dans cette étude proviennent d'une campagne de mesure réalisée sur une voie ferrée à Revigny durant le mois de février, à l'aide d'un chariot instrumenté (avec une vitesse maximale de 1.5m/s).

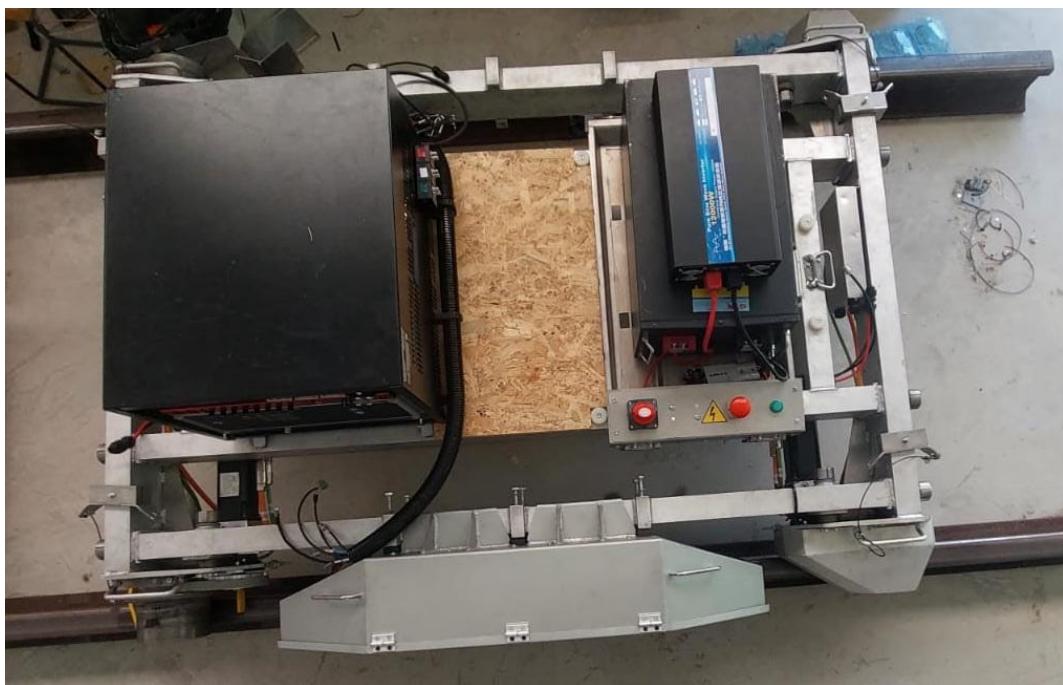


FIGURE 8 – Chariot instrumenté et système d'enregistrement de données

Ce chariot est équipé de plusieurs capteurs synchronisées.

- **4 microphones** : deux microphones omnidirectionnels et deux microphones cardioïdes orientés vers le contact rail-roue pour s'assurer de bien se focaliser sur le rail, positionnés à l'avant du chariot.
- **4 accéléromètres unidirectionnels** : deux mesurant les vibrations verticales et deux mesurant les vibrations horizontales.
- **1 accéléromètre triaxial** : placé au centre du chariot pour capter les vibrations dans les trois axes.
- **1 caméra linéaire** : utilisée pour capturer une image continue du rail.
- **1 Data Logger** : pour l'enregistrement centralisé des signaux.(voir Fig. 9).
- **1 application mobile** : utilisée pour assister les opérateurs dans la labellisation des événements détectés sur le terrain.



FIGURE 9 – Data logger et les channels utilisés

3.2 Description des données acquises

3.2.1 Présentation des fichiers MF4

Lors de la phase d’acquisition, plusieurs essais ont été réalisés et enregistrés sous forme de fichiers .mf4. Certains enregistrements de courte durée avaient uniquement pour objectif de tester le bon fonctionnement du chariot ou du datalogger, avant de lancer un enregistrement plus long.

Chaque fichier est automatiquement nommé en fonction de la date et de l’heure de début de l’acquisition, ce qui permet de garantir une traçabilité claire des données collectées. Le tableau ci-dessous illustre l’ensemble des fichiers générés,

Nom du fichier	Description
RecordFile_25-02-24_12_40_37_092.mf4	Aller – non utilisé
RecordFile_25-02-24_12_41_12_432.mf4	Aller – non utilisé
RecordFile_25-02-24_12_45_10_061.mf4	Aller – non utilisé
RecordFile_25-02-24_12_50_29_898.mf4	Aller – utilisé
RecordFile_25-02-24_13_09_59_044.mf4	Aller – non utilisé
RecordFile_25-02-24_13_19_22_003.mf4	Aller – utilisé
RecordFile_25-02-24_13_34_48_567.mf4	Retour – utilisé

TABLE 1 – Présentation des fichiers .mf4 avec leur utilisation dans les expériences

3 Présentation du dispositif expérimental et préparation des données

Parmi l'ensemble des fichiers .mf4 acquis, seuls trois ont été retenus pour les travaux présentés dans ce mémoire,

- un fichier principal représentant le **retour**, d'une durée d'environ 30 minutes ;
- deux fichiers correspondant à l'**aller**, dont la durée cumulée est d'environ 25 minutes.

Ces enregistrements de longue durée constituent la base de données principale utilisée pour les expérimentations, ils ont pu être lus en **Python** grâce à la bibliothèque **asammdf**, spécialement conçue pour la manipulation des formats .mf4.

Pour chaque fichier .mf4, plusieurs canaux sont enregistrés. Chaque canal est associé à un capteur spécifique. Le tableau ci-dessous présente les différents canaux et les capteurs auxquels ils sont reliés.

Canal	Capteur associé
Acc_1 Direct	Mesure des vibrations verticales côté droit
3A_Z Direct	Accéléromètre 3 axes – axe Z
Son_1 Direct	Microphone cardioïde côté droit
Son_2 Direct	Microphone omnidirectionnel côté droit
3A_Y Direct	Accéléromètre 3 axes – axe Y
3A_X Direct	Accéléromètre 3 axes – axe X
Acc_3 Direct	Mesure des vibrations verticales côté gauche
Son_4 Direct	Microphone cardioïde côté gauche
Son_3 Direct	Microphone omnidirectionnel côté gauche
Acc_4 Direct	Mesure des vibrations horizontales côté gauche
Chariot Direct	Signal carré, un pic toutes les 0,7 mm (utilisé pour calculer la vitesse)
Acc_2 Direct	Mesure des vibrations horizontales côté droit

TABLE 2 – Liste des canaux et capteurs associés pour chaque fichier .mf4.

D'après le signal **Chariot Direct**, il a été possible de calculer les vitesses lors des acquisitions des fichiers .mf4. Les figures ci-dessous présentent les vitesses calculées pour le fichier **aller utilisé** ainsi que pour le fichier **retour utilisé**.

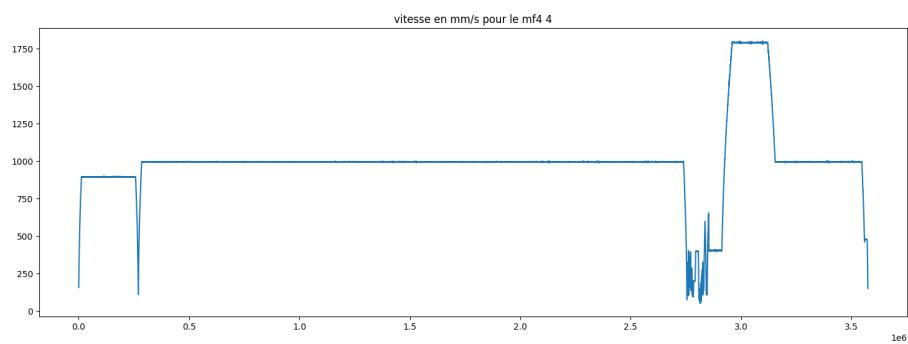


FIGURE 10 – Vitesse de l'Enregistrement de la phase de retour

3 Présentation du dispositif expérimental et préparation des données

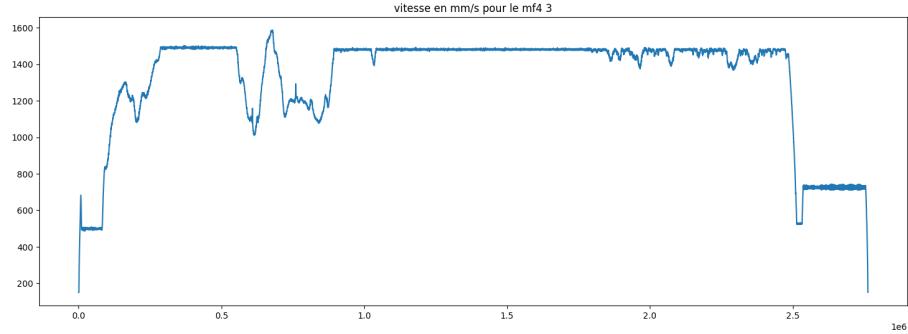


FIGURE 11 – Vitesse de l’Enregistrement de la phase d’aller

En plus des capteurs, une caméra linéaire a été installée à droite du chariot pour capturer des images de la voie. Chaque image possède une résolution de 2048×4096 pixels, avec un pas spatial de 0.175 mm par pixel. La figure ci-dessous illustre un exemple d’image linéaire contenant un joint.

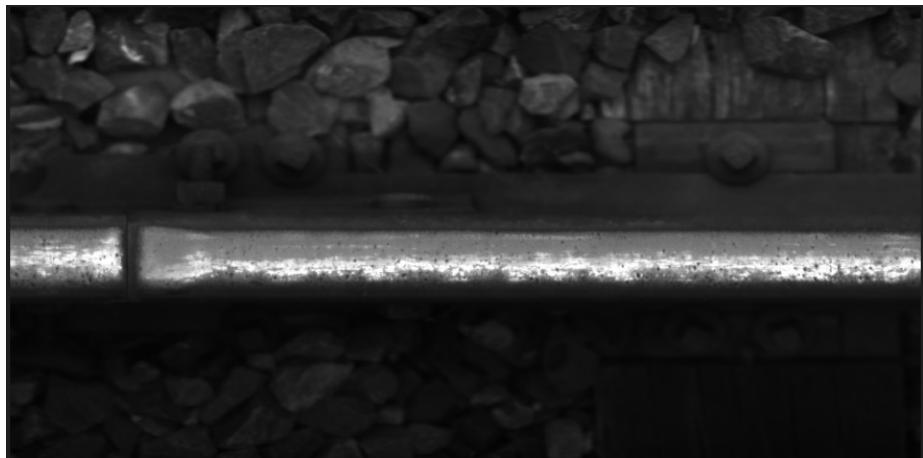


FIGURE 12 – Image linéaire représentant un joint.

Pour faciliter la labellisation des défauts, une application mobile a été développée permettant de réaliser des annotations manuelles. Les annotations sont enregistrées sous forme de fichiers texte, où l’utilisateur peut indiquer la présence d’un **joint** ou d’une **source inconnue** (défaut en général).

3.3 Synchronisation et annotation des données

Afin de tirer parti des différentes modalités, une étape de synchronisation des signaux est nécessaire. Celle-ci est réalisée à l’aide de scripts Python, afin d’aligner temporellement les données audio, vibratoires et visuelles.

3 Présentation du dispositif expérimental et préparation des données

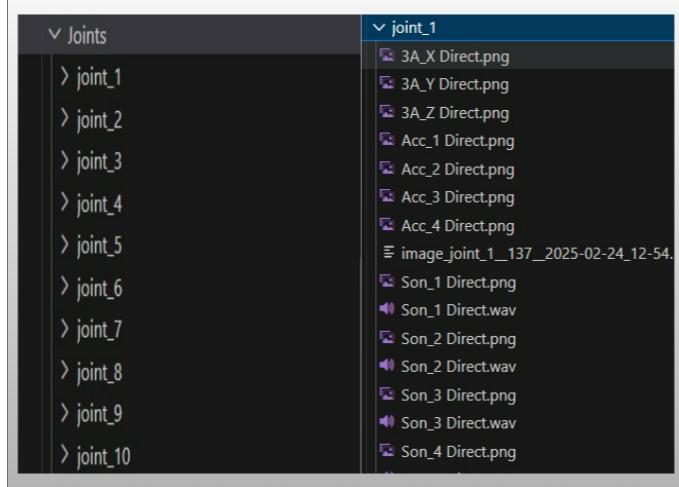


FIGURE 13 – Exemple d'un dossier synchronisé

Pour faciliter l'annotation des événements, l'outil **Audacity** a été utilisé pour visualiser les signaux audio et annoter précisément la position des joints. Ces annotations ont ensuite été converties en labels temporels communs pour toutes les modalités.

Les labels utilisés correspondent aux roues suivantes :

Abréviation	Description
avd	roue avant droite
avg	roue avant gauche
ard	roue arrière droite
arg	roue arrière gauche
avd+avg	deux roues avant (gauche et droite) passent sur le joint simultanément
ard+arg	deux roues arrière (gauche et droite) passent sur le joint simultanément

TABLE 3 – Abréviations des roues et combinaisons utilisées pour l'annotation

La figure ci-dessous présente deux illustrations complémentaires : la première montre un exemple d'annotation d'un signal audio réalisée avec *Audacity*, tandis que la seconde met en évidence, de manière détaillée, le passage d'un joint perçu successivement par les quatre roues.

3 Présentation du dispositif expérimental et préparation des données

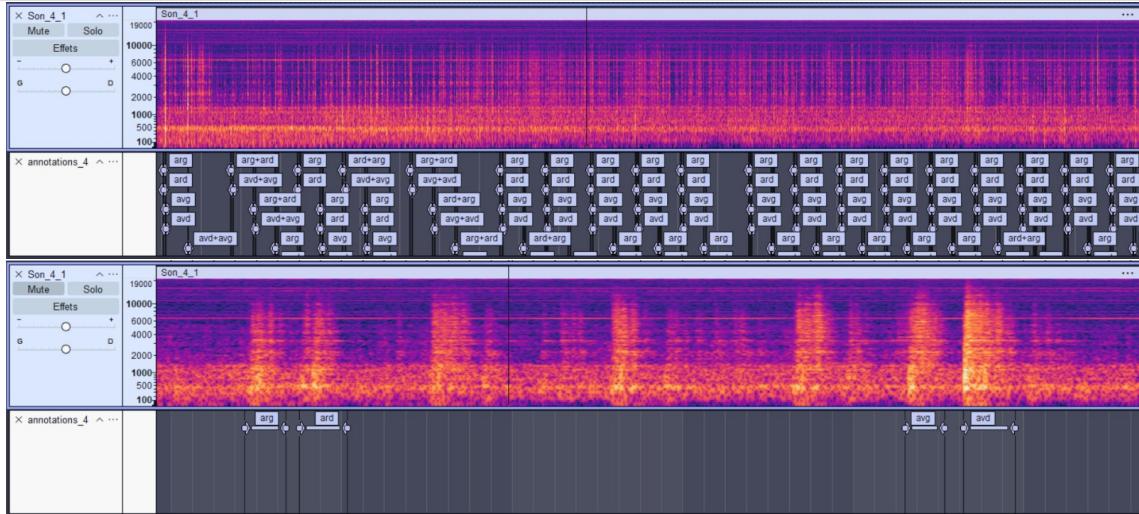


FIGURE 14 – Exemple de labellisation sur Audacity pour un signal acoustique

À partir des annotations de joints obtenues avec **Audacity**, ainsi que des informations enregistrées via l’application mobile utilisée lors de la journée d’acquisition, les défauts présents sur la voie ont pu être identifiés et annotés de manière précise. Ces deux sources complémentaires ont permis de localiser les défauts en corrélant les événements détectés avec les positions enregistrées sur le terrain.

4 Prétraitement et visualisation des données

4.1 Extraction des caractéristiques

Dans un premier temps, les modèles ont été entraînés sur la détection des joints. Pour cela, différentes représentations temporelles et fréquentielles ont été extraites à partir des signaux bruts afin d'enrichir l'information exploitable par les modèles d'apprentissage.

Plusieurs types de descripteurs ont été extraits à partir des signaux audio et vibration, en utilisant différentes bibliothèques Python adaptées à chaque type de traitement.

4.1.1 Transformée de Fourier (FFT)

calculée à l'aide de la bibliothèque `librosa`, elle permet de représenter le contenu fréquentiel global du signal. Mathématiquement, la FFT d'un signal $x[n]$ de taille N est donnée par :

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{kn}{N}}, \quad k = 0, 1, \dots, N - 1$$

où $X[k]$ représente la composante fréquentielle à la fréquence k , et j est l'unité imaginaire.

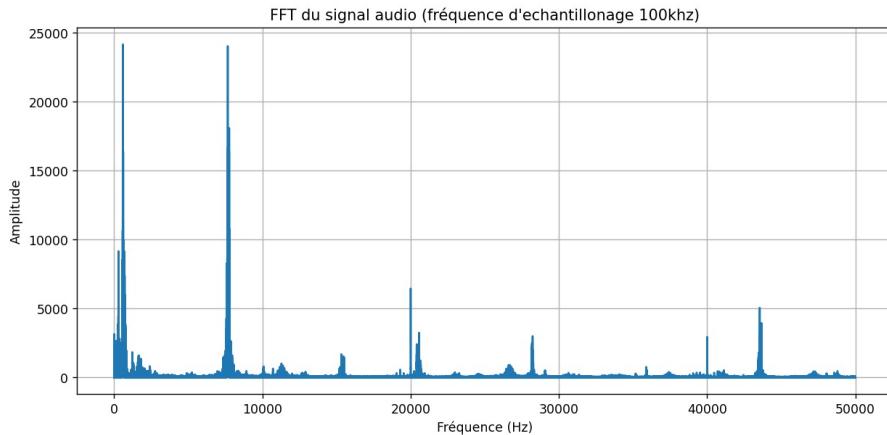


FIGURE 15 – Exemple de spectre de fréquence (FFT)

4.1.2 Spectrogramme classique

Le spectrogramme est une représentation temps-fréquence du signal audio obtenue via la Transformée de Fourier à court terme (STFT). Il permet de visualiser la répartition de l'énergie du signal en fonction du temps et des fréquences. Les

fréquences sont représentées de manière linéaire, ce qui peut rendre moins visibles les informations situées dans les basses fréquences, pourtant cruciales pour l'analyse audio.

4.1.3 Spectrogramme Mel

Également extrait via `librosa`, il s'appuie sur le spectrogramme classique mais applique un banc de filtres triangulaires répartis selon l'échelle de Mel. La relation entre la fréquence en Hz (f) et l'échelle de Mel (m) est donnée par :

$$m = 2595 \cdot \log_{10} \left(1 + \frac{f}{700} \right)$$

Contrairement au spectrogramme classique, cette transformation comprime les hautes fréquences et accorde davantage de résolution aux basses fréquences, ce qui correspond mieux à la perception auditive humaine. Cela en fait une représentation plus adaptée pour des tâches d'apprentissage automatique sur des signaux audio.

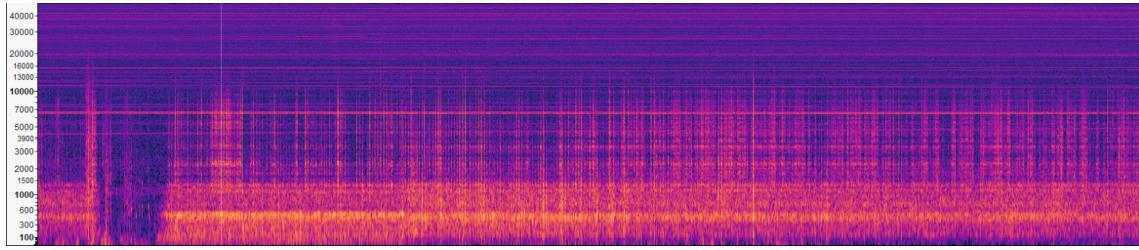


FIGURE 16 – Spectrogramme Mel du signal

D'après la figure 16., on observe un bruit constant autour de 6800 Hz, ainsi qu'aux alentours de 4000 Hz et dans les hautes fréquences au-delà de 20 000 Hz. Par ailleurs, les motifs présents ne dépassent pas une fréquence maximale d'environ 22 kHz. Cela m'a conduit à choisir une fréquence maximale de 22 kHz, ce qui correspond aussi à la limite de perception de l'oreille humaine.

4.1.4 Coefficients MFCC (Mel-Frequency Cepstral Coefficients)

obtenus via `librosa`, ils permettent de résumer les caractéristiques spectrales du signal en un petit nombre de coefficients. Les MFCC sont calculés en appliquant une transformation en cosinus discrète (DCT) aux log-énergies du spectrogramme Mel. Formellement, pour un vecteur de log-énergies S_k (dans la bande Mel), le $n^{\text{ième}}$ coefficient MFCC c_n est donné par :

$$c_n = \sum_{k=1}^K \log(S_k) \cdot \cos \left[\frac{\pi n}{K} \left(k - \frac{1}{2} \right) \right], \quad n = 0, 1, \dots, N - 1$$

où K est le nombre total de bandes Mel, et N le nombre de coefficients MFCC conservés.

4.1.5 Énergie du signal

L'**énergie du signal** dans une fenêtre temporelle de taille N est calculée comme la somme des carrés des coefficients spectraux (voir Fig. 17), y compris, par exemple, les amplitudes issues de la FFT :

$$E = \sum_{n=0}^{N-1} |X[n]|^2,$$

où $X[n]$ représente les coefficients spectraux associés à cette fenêtre.

Lien avec le domaine temporel – Théorème de Parseval Le calcul ci-dessus trouve son fondement théorique dans le **théorème de Parseval**, qui établit l'égalité entre l'énergie d'un signal mesurée dans le domaine temporel et dans le domaine fréquentiel. Dans sa forme continue, l'égalité de Parseval s'écrit :

$$\int_{-\infty}^{+\infty} |x(t)|^2 dt = \int_{-\infty}^{+\infty} |X(f)|^2 df,$$

où $x(t)$ est le signal temporel et $X(f)$ sa transformée de Fourier. Cela signifie que l'énergie totale du signal reste identique que l'on travaille dans le domaine temporel ou fréquentiel :

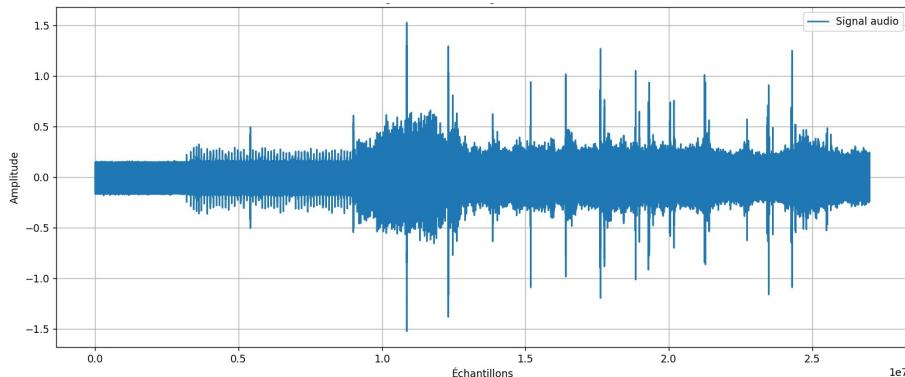


FIGURE 17 – Évolution de l'énergie du signal

4.1.6 Enveloppe du signal

obtenue à l'aide de la bibliothèque `scipy.signal` via le calcul de l'enveloppe par la transformée de Hilbert. Elle permet de suivre l'enveloppe temporelle du signal. Mathématiquement, l'enveloppe $e(t)$ d'un signal $x(t)$ est donnée par :

$$e(t) = \sqrt{x(t)^2 + \mathcal{H}\{x(t)\}^2}$$

où $\mathcal{H}\{x(t)\}$ est la transformée de Hilbert de $x(t)$.

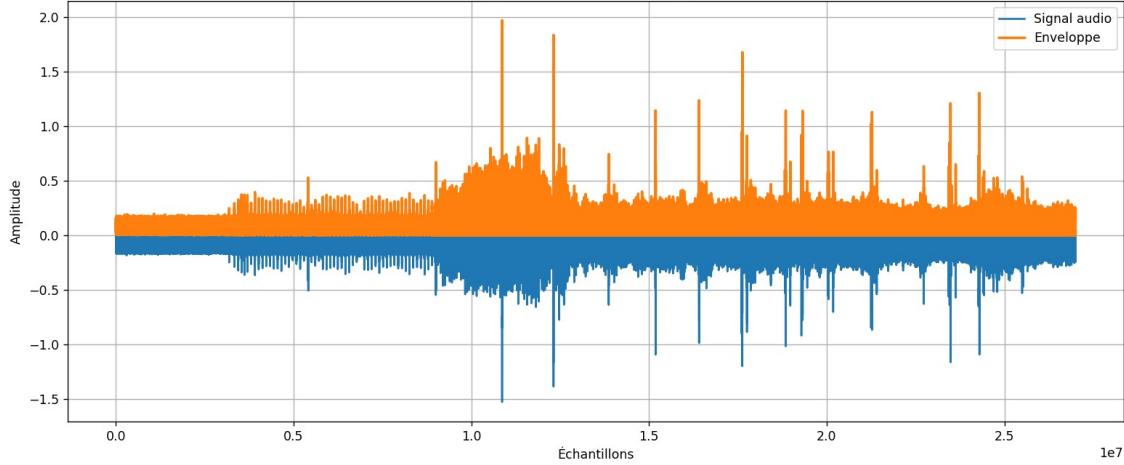


FIGURE 18 – Enveloppe du signal audio

4.2 Visualisation des données en fonction des labels

4.2.1 Analyse des Transformées de Fourier (FFT) en fonction des labels

Dans cette section, nous analysons les coefficients de la Transformée de Fourier (FFT) extraits des segments de signal, en les regroupant selon les labels annotés (présence ou absence de joint). Cette analyse permet d'étudier les variations spectrales caractéristiques des défauts détectés.

Pour chaque groupe, nous calculons des statistiques descriptives sur les coefficients FFT, telles que la moyenne, le maximum, le minimum, l'écart-type ainsi que la somme des amplitudes spectrales. Ces indicateurs sont ensuite corrélés avec les labels afin d'évaluer leur pertinence discriminative.

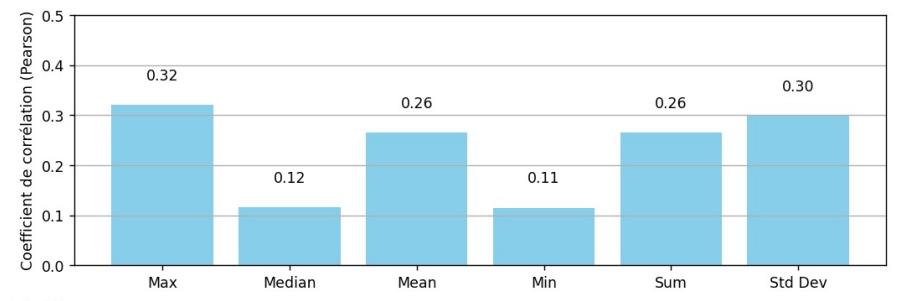


FIGURE 19 – Corrélations entre les statistiques extraites des coefficients FFT (moyenne, max, min, écart-type, somme) et les labels de défauts.

Les analyses statistiques des coefficients de la FFT, après normalisation par l'énergie du signal (division des coefficients spectraux par l'énergie correspondante), ont montré que, bien que certaines corrélations avec les labels puissent être observées, l'utilisation exclusive de ces coefficients pour la détection des joints demeure limitée. En effet, la forte présence de bruit dans les signaux complique la séparation nette entre les classes, réduisant ainsi la fiabilité de la FFT comme unique critère. Cette observation suggère non seulement la nécessité de combiner d'autres types de caractéristiques, mais également que l'énergie du signal elle-même pourrait constituer une information pertinente à exploiter.

4.2.2 Visualisation des spectrogrammes et de l'énergie

Afin de mieux comprendre la structure temporelle et fréquentielle des signaux, nous avons analysé les spectrogrammes ainsi que la distribution de l'énergie en fonction des labels. La figure ci dessous illustre un exemple de spectrogramme temporel accompagné en dessous de la représentation des labels et de la variation de l'énergie du signal.

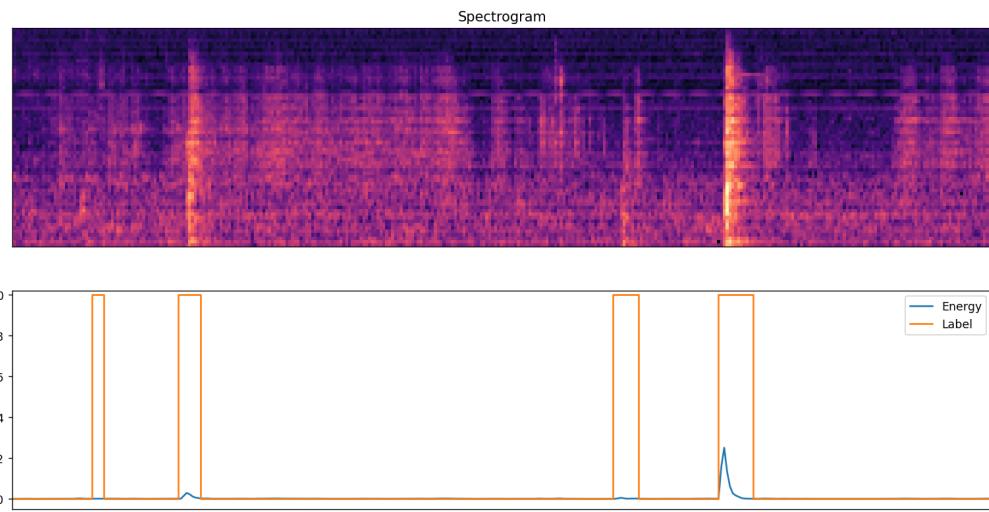


FIGURE 20 – Spécogramme et énergie en fonction des labelisations

Contrairement aux coefficients FFT seuls, dont la capacité à discriminer les joints était limitée à cause du bruit important, les spectrogrammes permettent une meilleure visualisation de l'évolution des fréquences dans le temps, ce qui est crucial pour détecter les événements transitoires caractéristiques des joints. De plus, l'analyse de l'énergie fournit une mesure complémentaire, reflétant l'intensité du signal dans les différentes fenêtres temporelles, ce qui peut contribuer à améliorer la détection des défauts.

4.3 Prétraitement des données pour l'apprentissage automatique

Cette section présente les différentes étapes de prétraitement des données mises en œuvre afin de préparer les données pour l'entraînement des modèles de machine learning.

4.3.1 Réduction de dimensionnalité par PCA

Étant donné la grande dimensionnalité des caractéristiques extraites, il est essentiel de recourir à une méthode de réduction de dimension afin de limiter la complexité computationnelle et d'éviter le surapprentissage. Pour cela, une analyse en composantes principales (PCA) est appliquée de manière à conserver la variance maximale des données tout en réduisant le nombre de dimensions. Chaque composante principale est ensuite normalisée en divisant par la racine carrée de la variance expliquée, permettant ainsi une meilleure homogénéisation des axes. Cependant, cette approche présente également certaines limites : en particulier, la PCA ne tient pas compte de la structure temporelle intrinsèque des signaux, ce qui peut conduire à une perte d'information pertinente liée à la dynamique temporelle.

4.3.2 Gestion du déséquilibre des classes

Pour pallier le déséquilibre des classes dans les données, des poids spécifiques sont calculés pour chaque classe, afin d'assurer une prise en compte équitable lors de l'entraînement des modèles.

4.3.3 Création d'échantillons contextuels

Afin d'enrichir les représentations, des échantillons contextuels sont générés en intégrant pour chaque fenêtre de données ses voisins immédiats. Cette méthode permet de capturer l'information locale autour d'un instant donné.

4.3.4 Construction de séquences pour modèles récurrents

Les données sont organisées en séquences temporelles compatibles avec les architectures de réseaux de neurones récurrents (RNN), pour tirer parti de la dimension temporelle des signaux.

4.3.5 Normalisation des données

Différentes techniques de normalisation sont testées lors de mes travaux :

- Normalisation Min-Max appliquée colonne par colonne (feature-wise)
- Normalisation Min-Max globale sur l'ensemble du dataset
- Standardisation (centrage et réduction)

5 Approche d'apprentissage automatique pour la détection des joints

5.1 Modélisation basée sur les signaux audio

5.1.1 Description des métriques d'évaluation

Pour évaluer les performances des modèles de machine learning appliqués à la détection des joints, plusieurs métriques seront utilisées, principalement focalisées sur la classe « joint » :

- **F1-score** de la classe joint : mesure harmonique entre la précision et le rappel, reflétant un compromis entre ces deux indicateurs.
- **Précision** (Precision) de la classe joint : proportion des prédictions positives correctes parmi toutes les prédictions positives.
- **Rappel** (Recall) de la classe joint : proportion des vrais positifs détectés parmi toutes les instances positives réelles.
- **Matrice de confusion** : tableau résumant les performances du modèle en termes de vrais positifs, faux positifs, vrais négatifs et faux négatifs.
- **Courbe ROC** (Receiver Operating Characteristic) : courbe représentant le compromis entre le taux de vrais positifs et le taux de faux positifs à différents seuils de décision.

Dans la suite, l'entraînement des modèles sera réalisé sur les données de retour (30 minutes), divisées en trois sous-ensembles : train, validation et test. Cette division permettra de sélectionner la meilleure architecture en fonction de la convergence du modèle et de sa capacité de généralisation. Enfin, la performance finale sera évaluée sur les données d'aller, qui n'ont pas été utilisées pendant l'entraînement.

5.1.2 Essai avec des modèles de machine learning classiques

Dans cette partie, nous avons utilisé le modèle *XGBoost* appliqué sur différentes caractéristiques extraites des signaux audio, à savoir les coefficients FFT, Mel et MFCC. L'objectif principal était d'évaluer la pertinence de ces différentes représentations pour la détection des joints, en sélectionnant les caractéristiques les plus discriminantes à utiliser dans la suite des traitements.

5.1.3 Optimisation des hyperparamètres avec Optuna

L'optimisation des hyperparamètres est une étape cruciale dans l'entraînement des modèles d'apprentissage automatique. En effet, le choix de paramètres tels que le taux d'apprentissage, la profondeur d'un réseau, ou encore la taille des batchs peut fortement influencer les performances finales du modèle.

Parmi les outils récents d'optimisation automatique, Optuna s'est imposé comme une librairie efficace, flexible et adaptée aux problèmes réels. Son principe repose sur

une recherche guidée par un algorithme bayésien appelé **TPE** (**T**ree-**s**tructured **P**arzen **E**stimator) (voir Fig. 21).[2].

Principe du TPE. Contrairement à une recherche exhaustive comme le *Grid Search*, ou aléatoire comme le *Random Search*, le TPE modélise la fonction objectif de manière probabiliste. L'idée est de séparer les essais en deux groupes :

- les essais prometteurs (valeurs de la fonction objectif faibles, donc proches de l'optimum),
- les essais non prometteurs.

À partir de cette séparation, le TPE construit deux modèles de distribution de probabilité (souvent des estimateurs de densité basés sur des noyaux). La recherche de nouveaux hyperparamètres se fait en maximisant le rapport de vraisemblance entre la distribution des valeurs prometteuses et celle des moins prometteuses. Autrement dit, les futurs essais sont plus concentrés dans les zones de l'espace des hyperparamètres qui ont montré de bons résultats.

Contexte d'utilisation. Optuna est particulièrement adapté dans les contextes suivants :

- modèles complexes ayant un grand nombre d'hyperparamètres (réseaux de neurones profonds, gradient boosting, etc.) ;
- problèmes où chaque évaluation du modèle est coûteuse (temps de calcul long) ;
- recherche adaptative où l'on souhaite stopper automatiquement les essais peu prometteurs grâce à la fonctionnalité de *pruning*.

Comparaison avec d'autres méthodes. Plusieurs alternatives existent pour l'optimisation des hyperparamètres :

- **Grid Search** : exploration systématique de toutes les combinaisons possibles. Très simple mais coûteux et inefficace lorsque le nombre de paramètres est élevé (effet d'explosion combinatoire).
- **Random Search** : sélection aléatoire des hyperparamètres. Plus efficace que Grid Search, mais sans exploitation des résultats passés.
- **Bayesian Optimization (ex. Spearmint, Hyperopt)** : repose sur un modèle probabiliste (souvent un processus gaussien). Très précis mais difficile à appliquer lorsque le nombre de dimensions est élevé.

Optuna, en utilisant le TPE, combine les avantages de la recherche bayésienne (exploitation des résultats passés) avec une grande flexibilité et scalabilité. De plus, son intégration simple avec des frameworks comme PyTorch, TensorFlow ou LightGBM, ainsi que ses mécanismes de *pruning*, en font une solution particulièrement efficace.

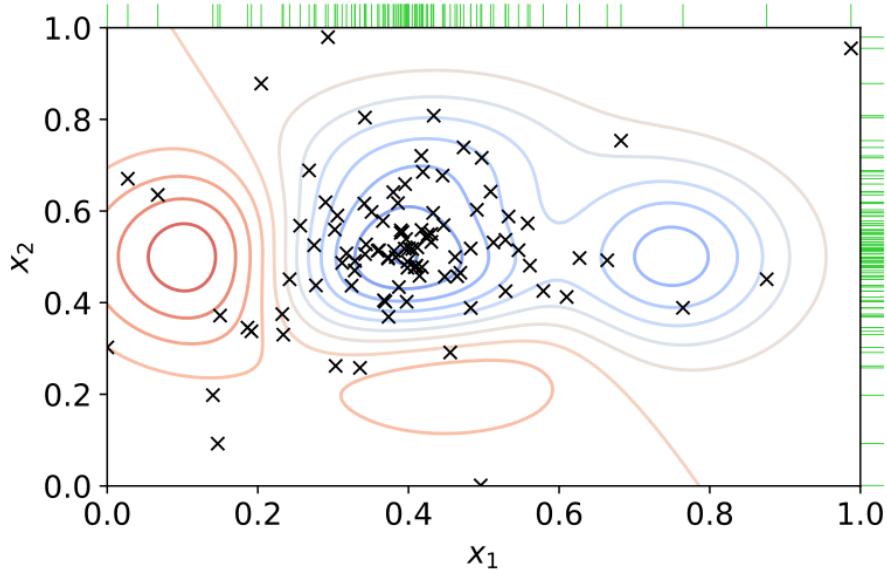


FIGURE 21 – Illustration du fonctionnement de l'algorithme TPE utilisé par Optuna dans un problème 2D.

5.1.4 Prétraitements et paramètres des données

Avant l'entraînement des modèles, un ensemble de paramètres a été fixé pour assurer la cohérence et la qualité des données. Le microphone utilisé pour les expériences est le **microphone cardioïde situé à droite (micro 1)**, captant principalement les sons émis dans la direction de la roue. Chaque enregistrement est découpé en fenêtres de **20 ms**, correspondant à **2048 échantillons** par fenêtre à une fréquence d'échantillonnage de **100kHz**, avec une fréquence maximale exploitée limitée à **22 kHz**.

Un **chevauchement de 50%** est appliqué entre les fenêtres lors du calcul des FFTs pour améliorer la résolution temporelle. Avant l'extraction des caractéristiques fréquentielles, la **moyenne globale du signal est soustraite**, suivie de la **sous-traction de la moyenne locale** sur chaque fenêtre afin de centrer les données.

Pour la réduction de dimensionnalité, une **Analyse en Composantes Principales (PCA)** est appliquée en conservant **95% de la variance totale**, permettant de réduire efficacement la redondance tout en préservant l'information pertinente.

5.1.5 Optimisation d'XGBoost avec Optuna

Dans le cadre de l'évaluation des caractéristiques (FFT, spectrogrammes Mel et MFCC), le modèle **XGBoost** a été utilisé. L'objectif est ici d'identifier les représentations les plus pertinentes pour la tâche de classification. Pour cela, une **optimisation bayésienne via Optuna** a été réalisée, permettant d'explorer efficacement l'espace des hyperparamètres.

Cinq hyperparamètres ont été sélectionnés comme variables d'optimisation, considérés comme ayant un impact majeur sur les performances du modèle :

- `learning_rate` : contrôle la vitesse d'apprentissage.
- `n_estimators` : nombre d'arbres dans l'ensemble.
- `max_depth` : profondeur maximale des arbres.
- `subsample` : proportion d'échantillons utilisée pour chaque arbre.
- `colsample_bytree` : proportion de colonnes (features) utilisée pour chaque arbre.

Ces paramètres ont été choisis pour leur influence directe sur la f1 score de la classe positive sur les données de test.

Analyse des meilleurs résultats obtenus avec XGBoost Une fois l'entraînement des modèles XGBoost terminé à l'aide d'Optuna, nous avons évalué les performances obtenues selon différents aspects de prétraitement et d'extraction des caractéristiques. Les résultats les plus pertinents sont présentés dans les figures suivantes :

- **Comparaison des caractéristiques** : Le graphique suivant montre les performances de XGBoost en fonction du type de représentation des signaux audio (FFT, Melspectrogrammes, MFCC). Cette étape nous a permis de sélectionner les caractéristiques les plus informatives.

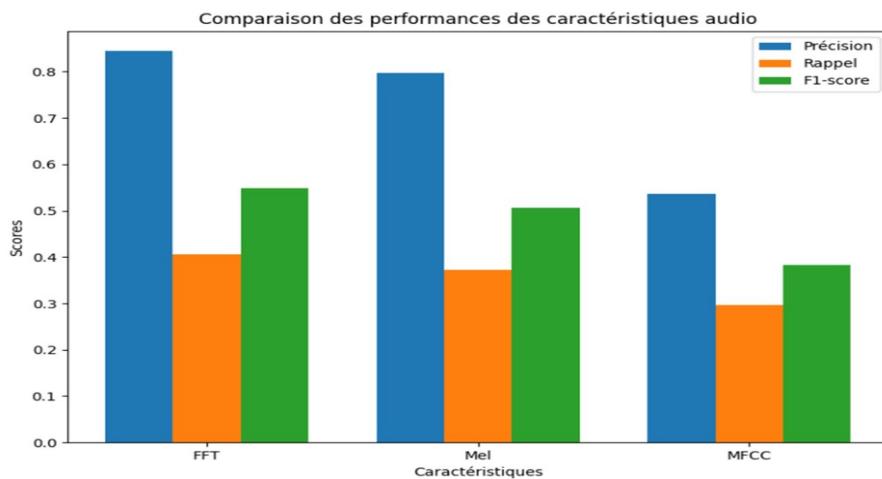


FIGURE 22 – Comparaison des performances de XGBoost selon les caractéristiques audio utilisées

- **Impact de la réduction de dimension** : Le graphique suivant présente l'effet de la réduction de dimension par PCA (avec et sans normalisation). L'objectif ici est de vérifier si cette étape améliore la généralisation du modèle.

5 Approche d'apprentissage automatique pour la détection des joints

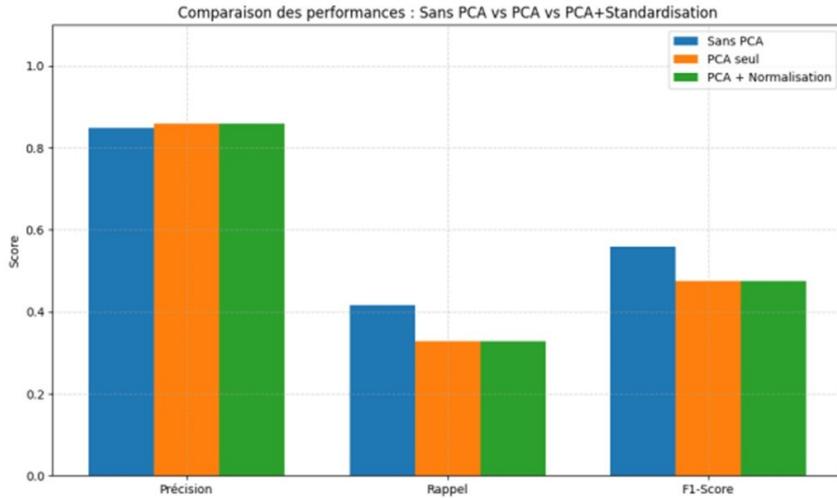


FIGURE 23 – Impact de la réduction de dimension par PCA sur les performances

- **Équilibrage des classes** : Le graphique ci-dessous montre l'influence de l'équilibrage des classes sur le F1-score de la classe minoritaire, ce qui est crucial pour notre tâche de classification binaire déséquilibrée.

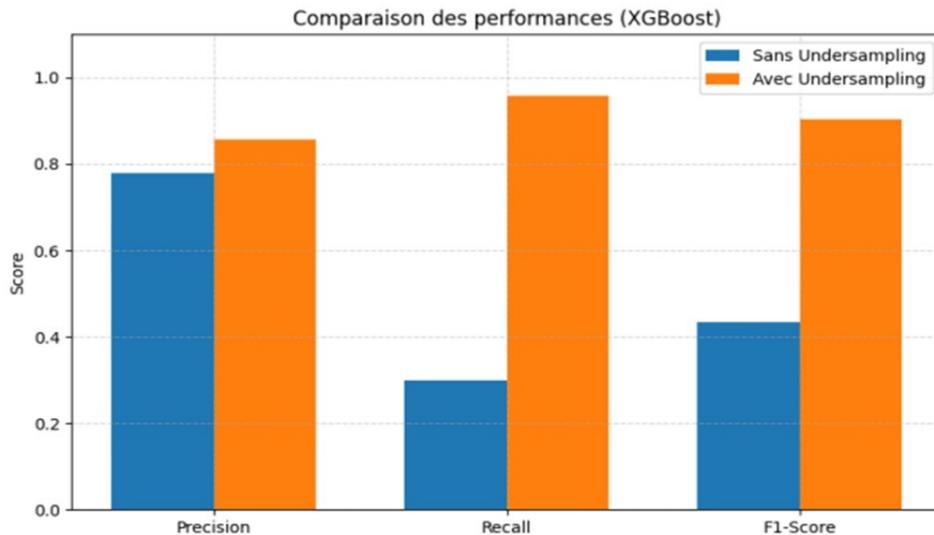


FIGURE 24 – Effet de l'équilibrage des classes sur les performances du modèle

Ces analyses nous ont permis de mieux comprendre l'impact de chaque étape de prétraitement sur la performance globale du modèle. Toutefois, une limite importante de ces approches classiques réside dans leur incapacité à exploiter la dynamique temporelle du signal.

Dans la suite de ce travail, nous explorons donc l'utilisation de réseaux de neurones récurrents de type LSTM, afin d'intégrer la dimension temporelle du signal audio dans le processus de classification.

5.1.6 Entraînement avec des réseaux de neurones récurrents (RNN)

Configuration initiale et protocoles d'expérimentation Dans un premier temps, plusieurs essais ont été menés afin d'évaluer l'impact de chaque hyperparamètre jugé important pour l'apprentissage du modèle RNN. L'architecture de base adoptée repose sur deux couches Bidirectional LSTM avec respectivement 128 et 64 unités, suivies de deux couches Dense de 64 puis 32 neurones, et d'une sortie avec une activation **sigmoïde**.

Avant l'entraînement, les données ont été prétraitées comme suit :

- Extraction des coefficients Mels à partir des FFTs avec un chevauchement de 50% ;
- Réduction de dimensionnalité par ACP (PCA) en conservant 95% de la variance ;
- Normalisation MinMax sur l'ensemble du dataset ;
- Calcul des poids de classes pour compenser le déséquilibre des labels ;
- Utilisation de la fonction de perte **Focal Loss** adaptée aux classes déséquilibrées ;
- Création de séquences de 40 observations par échantillon, avec un chevauchement de 50%.

L'entraînement est encadré par un mécanisme de *early stopping* avec une patience de 8 époques, basé sur l'évolution du **val_f1_score**.

Les expériences suivantes ont été menées pour analyser l'influence de différents paramètres :

1. **Impact du nombre de coefficients Mels** Dans les figures ci-dessous, nous illustrons l'impact du paramètre **nombre de coefficients cepstraux de type Mel (MFCCs)** extraits à partir des signaux audio. L'extraction a été réalisée en utilisant une fenêtre de **20 ms** avec un **chevauchement de 10 ms**. Trois configurations différentes ont été testées :
 - 16 coefficients,
 - 32 coefficients,
 - 64 coefficients.

Cela permet d'analyser comment la richesse spectrale influence la représentation du signal et les performances des modèles d'apprentissage.

5 Approche d'apprentissage automatique pour la détection des joints

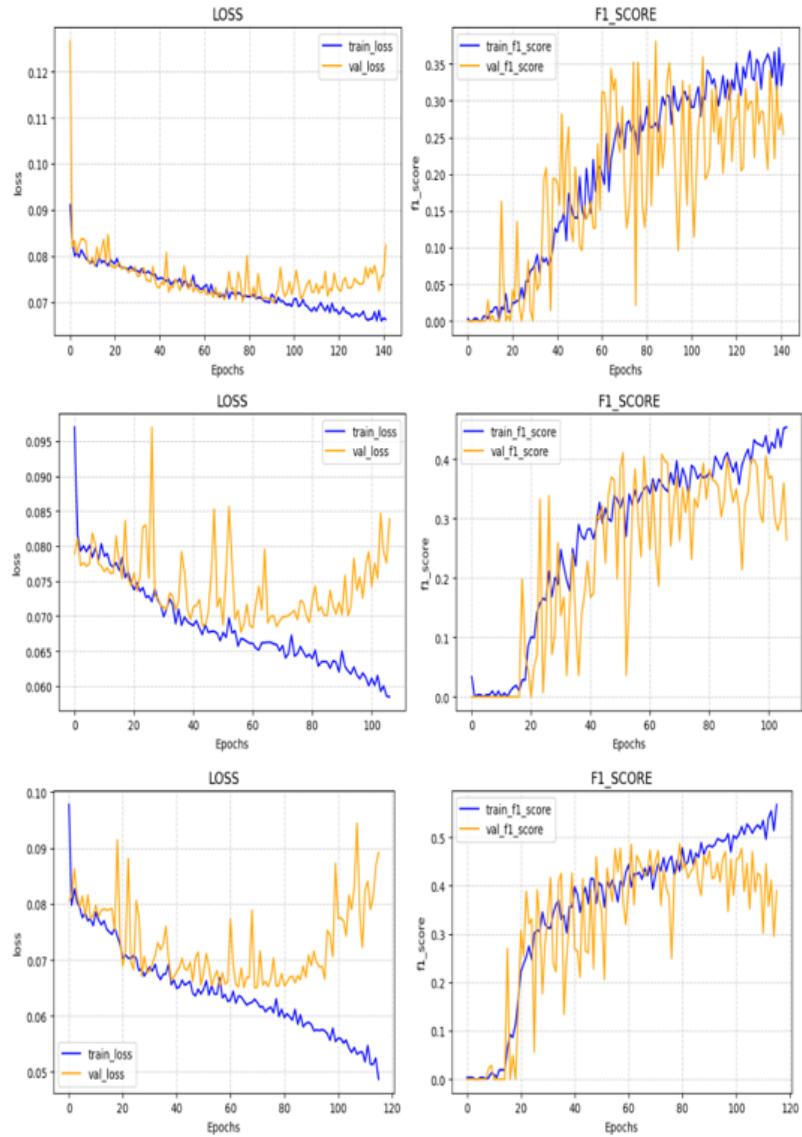


FIGURE 25 – Courbes d'entraînement et de validation selon le nombre de mels utilisé.

En conclusion, les expériences menées sur l'impact du nombre de coefficients Mels montrent qu'un plus grand nombre de coefficients permet une meilleure convergence du modèle. Cela peut s'expliquer par le fait qu'un nombre élevé de coefficients capture plus finement les détails fréquentiels du signal audio, offrant ainsi au réseau une représentation plus riche et discriminante. Toutefois, au-delà d'un certain seuil, l'augmentation du nombre de coefficients peut accroître la complexité et le risque de sur-apprentissage. Dans la suite des travaux, un compromis a été retenu avec **64 coefficients Mels**.

2. Impact de la fréquence maximale considérée lors du calcul des FFTs

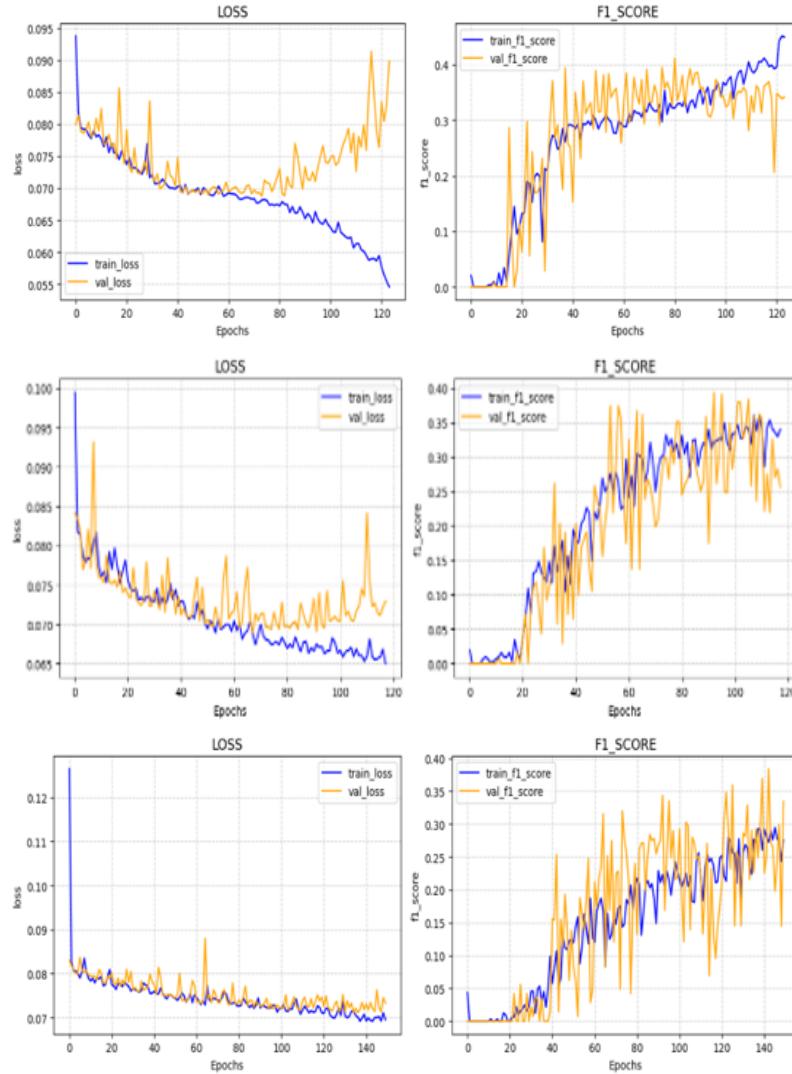


FIGURE 26 – Courbes d'entraînement et de validation selon la fréquence maximale utilisée.

Les analyses, appuyées à la fois par les résultats des modèles et par des observations sous *Audacity*, ont montré que les basses fréquences permettent de distinguer clairement les joints, mais uniquement dans des conditions calmes, sans bruit ambiant notable ce qui conduit à un sur-apprentissage qui est présenté dans les deux premières figures. En revanche, lorsque du bruit est présent — par exemple lors de l'application de scotch simulant l'effet d'un méplat sur la roue — les différences les plus marquées apparaissent dans les hautes fréquences. Ainsi, afin de capturer efficacement ces signatures fréquentielles liées aux défauts, il a été décidé de poursuivre les expérimentations avec

une **fréquence maximale de 22 kHz**, garantissant une couverture optimale des composantes haute fréquence et une convergence stable.

3. Influence du batch size sur la convergence du modèle

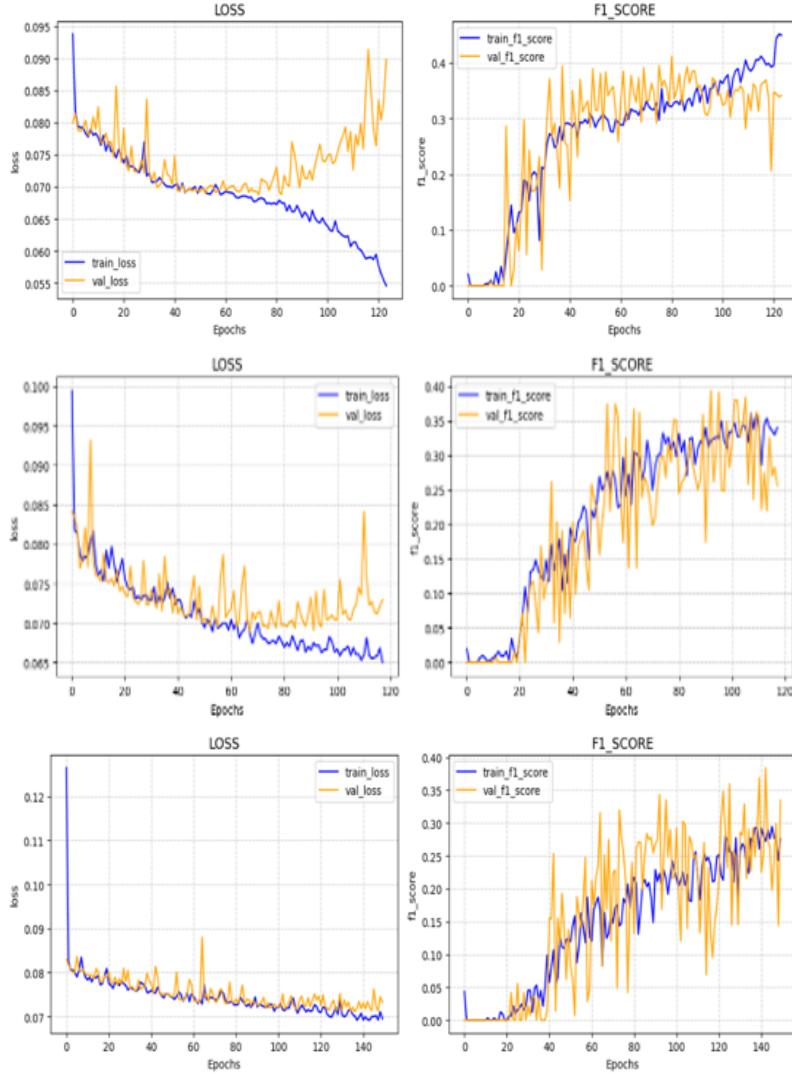


FIGURE 27 – Courbes d'entraînement et de validation selon la taille du batch utilisée.

Des expérimentations ont été réalisées avec différentes tailles de batch afin d'évaluer leur impact sur la convergence, le temps d'entraînement et la capacité de généralisation. Les petites tailles de batch (par exemple 16 et 32) ont montré un sur-apprentissage qui est marqué dans les courbe de loss, et au prix d'un temps d'entraînement nettement plus long. À l'inverse, des tailles de batch très élevées ont provoqué des problèmes matériels (mémoire GPU/CPU

insuffisante) et ont pénalisé la performance en présence d'un fort déséquilibre des classes. Ainsi, le compromis retenu a été d'adopter une taille de batch de 128, offrant un bon équilibre entre performance, temps d'entraînement et contraintes matérielles.

Pour optimiser le processus, il a été décidé d'adopter une stratégie d'ajustement dynamique du **learning rate** : l'augmenter au début de l'entraînement pour accélérer la convergence initiale, puis le réduire progressivement afin de stabiliser l'optimisation et d'éviter un sur-apprentissage. Cette approche a été mise en œuvre via la technique *Exponential Decay*.

Tentatives complémentaires non concluantes Dans le but d'augmenter les performances de détection, plusieurs approches supplémentaires ont été explorées, sans résultats probants :

- **Génération d'exemples positifs à l'aide de réseaux GAN** : les *Generative Adversarial Networks* (GAN) sont des réseaux génératifs composés de deux modules : un générateur, qui tente de produire des données réalistes, et un discriminateur, chargé de distinguer les vraies données des fausses. L'objectif était ici d'augmenter artificiellement le nombre d'exemples positifs. L'entraînement semblait converger correctement, mais les exemples générés ne se généralisaient pas bien et dégradaient la performance finale du modèle.
- **Utilisation d'OpenL3 pour l'extraction de caractéristiques audio** : OpenL3 est un modèle d'extraction de représentations audio pré-entraîné, basé sur l'apprentissage multimodal. Cependant, cet outil impose une taille de fenêtre fixe de **960 ms** pour les entrées audio, ce qui est relativement long dans le contexte de notre problème. Cette contrainte réduit fortement le nombre d'exemples positifs disponibles, rendant l'entraînement moins efficace et entraînant un déséquilibre aggravé entre les classes.
- **Optimisation des hyperparamètres via Optuna** : Une autre tentative consistait à utiliser la bibliothèque *Optuna* pour optimiser les hyperparamètres du modèle LSTM. Cependant, chaque essai (trial) était limité à seulement 20 époques pour des raisons de temps de calcul, ce qui s'est avéré insuffisant pour évaluer correctement les performances. Même en essayant d'accélérer la convergence (par exemple via un taux d'apprentissage élevé), la qualité des résultats restait médiocre. Augmenter significativement le nombre d'époques aurait rendu le processus trop coûteux temporellement. Pour cette raison, l'optimisation manuelle des hyperparamètres a été privilégiée par la suite.

Conclusion Comme l'ont montré les différentes courbes, les performances obtenues à ce stade restent très faibles. Face à ce constat, il a été décidé d'explorer deux axes d'amélioration : d'une part, la modification et l'enrichissement des caractéristiques extraites des données, et d'autre part, l'adaptation de l'architecture du réseau

de neurones afin de tenter d'optimiser les résultats.

5.1.7 Entraînement avec des réseaux de neurones récurrents convolutionnels (C-RNN)

Description détaillée de l'architecture et des données L'architecture proposée pour la détection des joints à partir des séquences temporelles est composée de plusieurs blocs complémentaires permettant d'extraire et de traiter efficacement les informations des signaux :

- **Couches de convolution 2D :**

Ces couches sont responsables de l'extraction automatique de caractéristiques locales dans les données. Chaque convolution est appliquée *uniquement sur l'axe fréquentiel*, ce qui permet de réduire la dimensionnalité des caractéristiques tout en préservant la structure temporelle. Les fonctions d'activation utilisées sont des LeakyReLU, qui permettent de garder un flux d'information même pour les valeurs négatives, évitant le problème de neurones morts.

- **Flattening :**

Après les convolutions, les sorties sont *aplatises* pour transformer la représentation en une forme compatible avec les couches LSTM. La nouvelle représentation a la forme ($n_samples, n_steps, n_features \times n_filters$).

- **Couches LSTM bidirectionnelles :**

Les LSTM (Long Short-Term Memory) permettent de capturer les dépendances temporelles passées et futures dans les séquences grâce à leur structure interne de mémoire. L'utilisation de *bidirectional LSTM* permet au réseau de prendre en compte à la fois le contexte avant et après chaque pas temporel, améliorant la capacité du modèle à détecter des événements courts comme les joints. Ces LSTM utilisent également une activation LeakyReLU pour mieux gérer les variations du signal et améliorer la convergence pendant l'entraînement.

- **Sortie du réseau :**

Le réseau retourne une annotation par séquence, c'est-à-dire qu'il prédit si la séquence contient ou non un joint, en cohérence avec la labellisation stricte utilisée lors de la création des séquences.

Model: "Detection_Joints"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 20, 130)	0
reshape (Reshape)	(None, 20, 130, 1)	0
conv2D_0 (Conv2D)	(None, 20, 130, 64)	1,024
leaky_re_lu (LeakyReLU)	(None, 20, 130, 64)	0
dropout (Dropout)	(None, 20, 130, 64)	0
conv2D_1 (Conv2D)	(None, 20, 65, 128)	73,856
leaky_re_lu_1 (LeakyReLU)	(None, 20, 65, 128)	0
dropout_1 (Dropout)	(None, 20, 65, 128)	0
reshape_1 (Reshape)	(None, 20, 8320)	0
bidirectional (Bidirectional)	(None, 20, 256)	8,651,776
leaky_re_lu_2 (LeakyReLU)	(None, 20, 256)	0
dropout_2 (Dropout)	(None, 20, 256)	0
bidirectional_1 (Bidirectional)	(None, 128)	164,352
leaky_re_lu_3 (LeakyReLU)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense (Dense)	(None, 2)	258

Total params: 8,891,266 (33.92 MB)

Trainable params: 8,891,266 (33.92 MB)

FIGURE 28 – Résumé de l'architecture du modèle C-RNN (`model.summary()`).

Concernant les données d'entrée, les coefficients Mels (64 coefficients) sont calculés sur une bande passante allant jusqu'à 22 kHz. Les dérivées première et seconde de ces coefficients sont ajoutées, en retirant l'énergie initiale mais en conservant et ajoutant des colonnes dédiées à l'énergie et à sa dérivée.

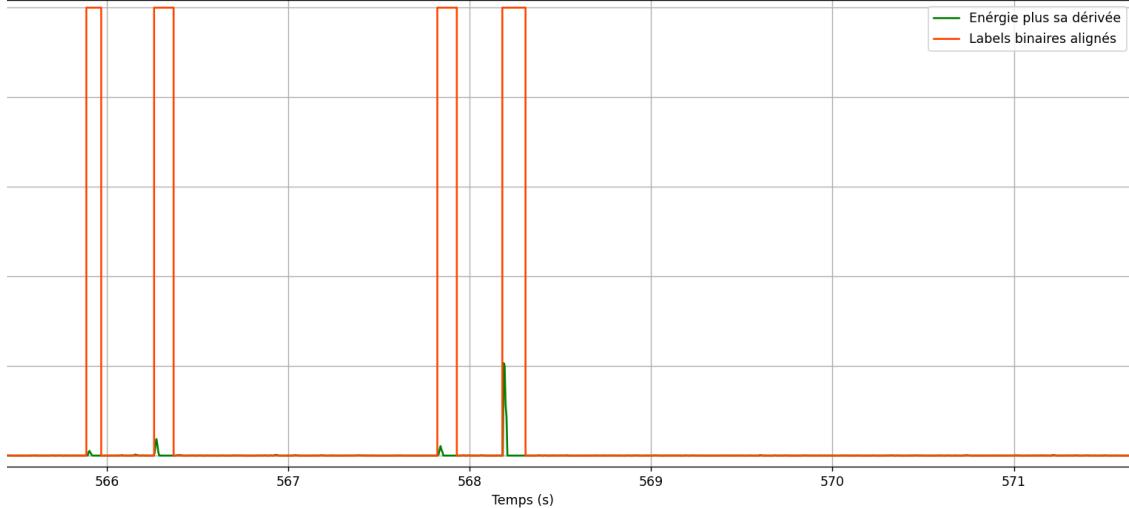


FIGURE 29 – L'énergie + sa dérivé en fonction des labels

La segmentation est réalisée en séquences de 20 trames, chaque trame correspondant à une fenêtre de 20 ms avec un chevauchement de 50 % (soit un déplacement de 10 ms) lors du calcul des coefficients de Mel. Ainsi, chaque séquence couvre une durée totale de 200 ms. La stratégie de labellisation adoptée reste stricte : une séquence est annotée « joint » dès lors qu'au moins une trame correspond à un joint. Comme la durée maximale d'un joint observée dans les données est d'environ 110 ms, cette configuration assure une capture adéquate de l'événement. De plus, aucune réduction de dimension par PCA n'a été appliquée, les tests ayant montré que cette technique altérait la structure temporelle avant la formation des séquences.

Résultats d'entraînement avec les micros individuels et fusion tardive des capteurs :

Nous avons entraîné le modèle C-RNN sur les données issues de chaque microphone individuel, ainsi que sur une fusion tardive combinant les sorties des quatre micros (micro cardioïde droit, deux micros omnidirectionnels à droite et à gauche, et micro cardioïde gauche).

Pour chaque microphone, nous présentons les résultats suivants :

- La courbe d'entraînement (perte et/ou précision selon l'entraînement),
- La matrice de confusion sur les données de test,

Microphone 1 (cardioïde à droite)

Le microphone a été utilisé pour la recherche des hyperparamètres du réseau lors de la phase d'entraînement. Une fois cette étape réalisée, la même architecture a ensuite été appliquée à l'entraînement avec les autres capteurs. Les résultats obtenus sont présentés ci-après.

5 Approche d'apprentissage automatique pour la détection des joints

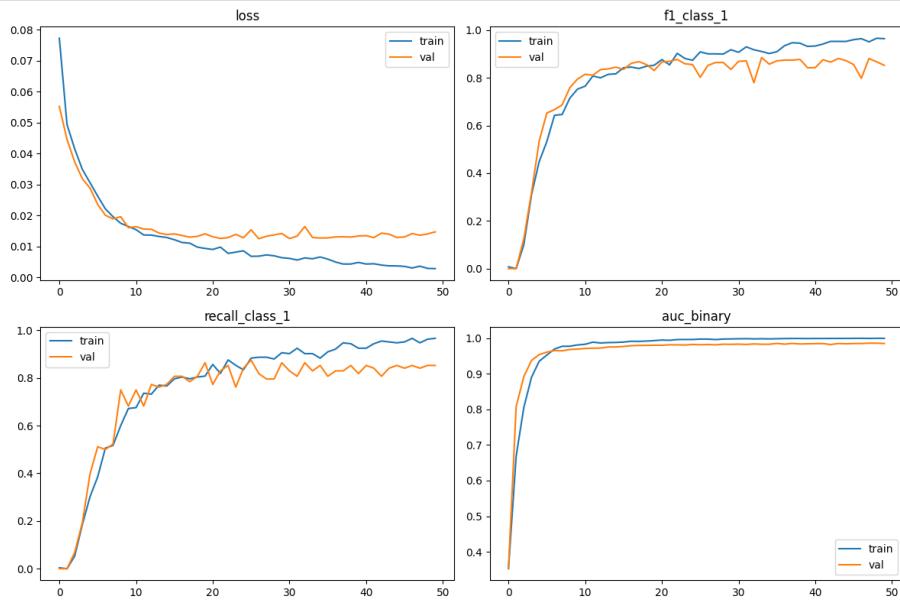


FIGURE 30 – Courbe d'entraînement pour le micro 1.

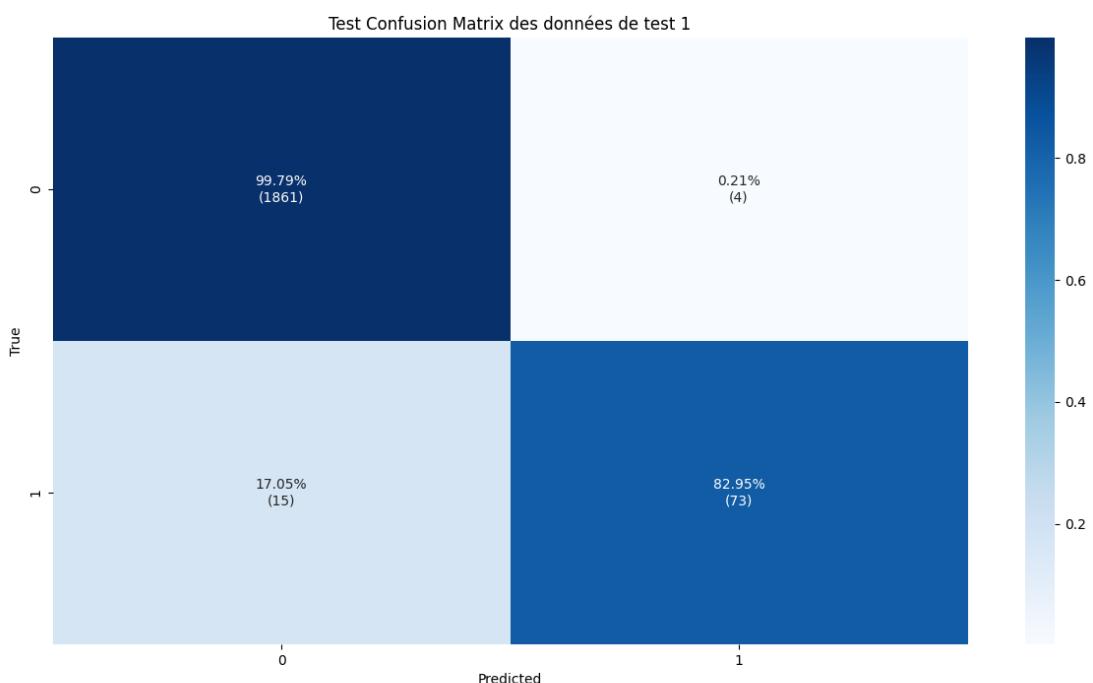


FIGURE 31 – Matrice de confusion sur les données de test pour le micro 1.

Microphone 2 (omnidirectionnel à droite)

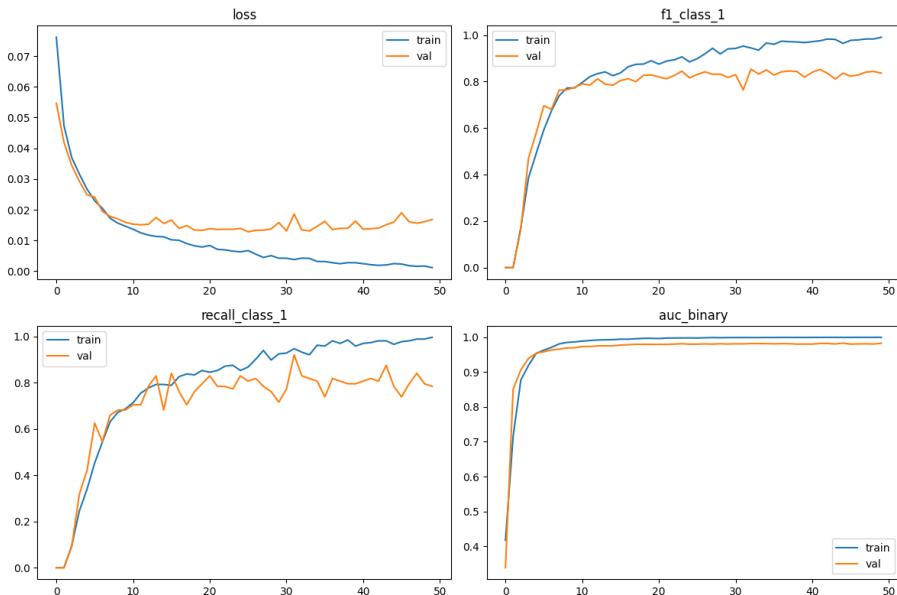


FIGURE 32 – Courbe d'entraînement pour le micro 2.

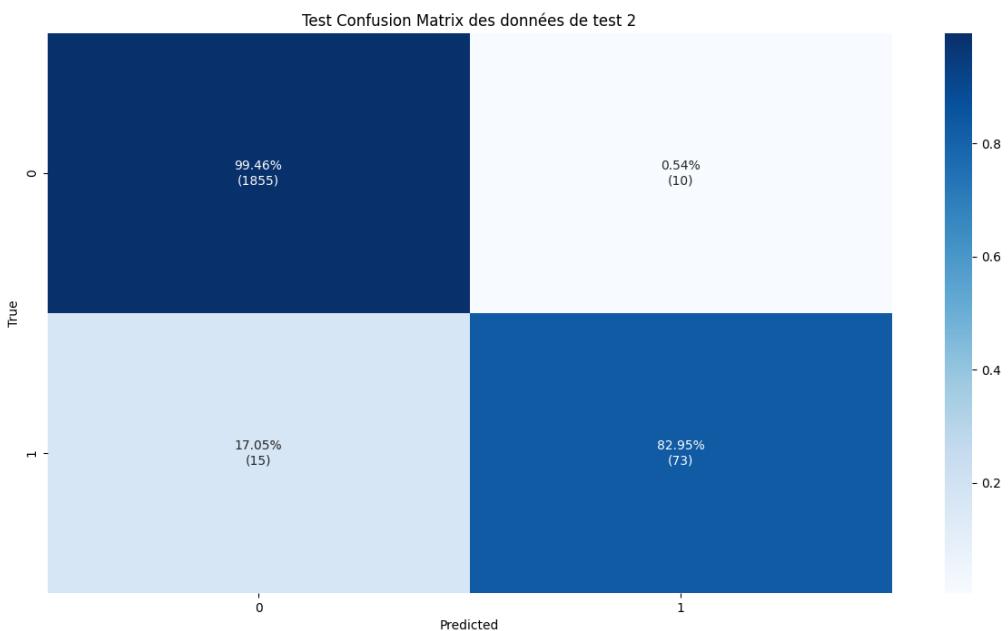


FIGURE 33 – Matrice de confusion sur les données de test pour le micro 2.

Microphone 3 (omnidirectionnel à gauche)

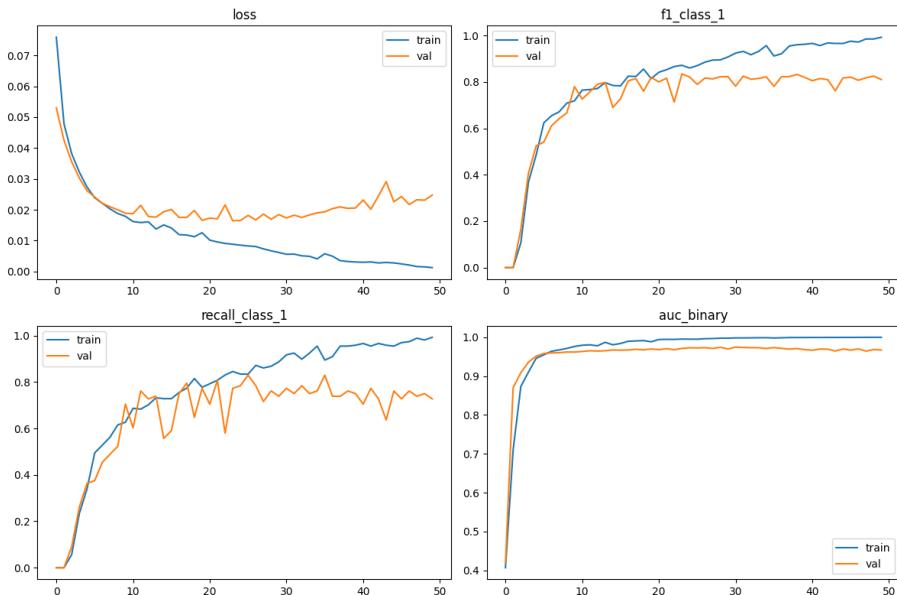


FIGURE 34 – Courbe d'entraînement pour le micro 3.

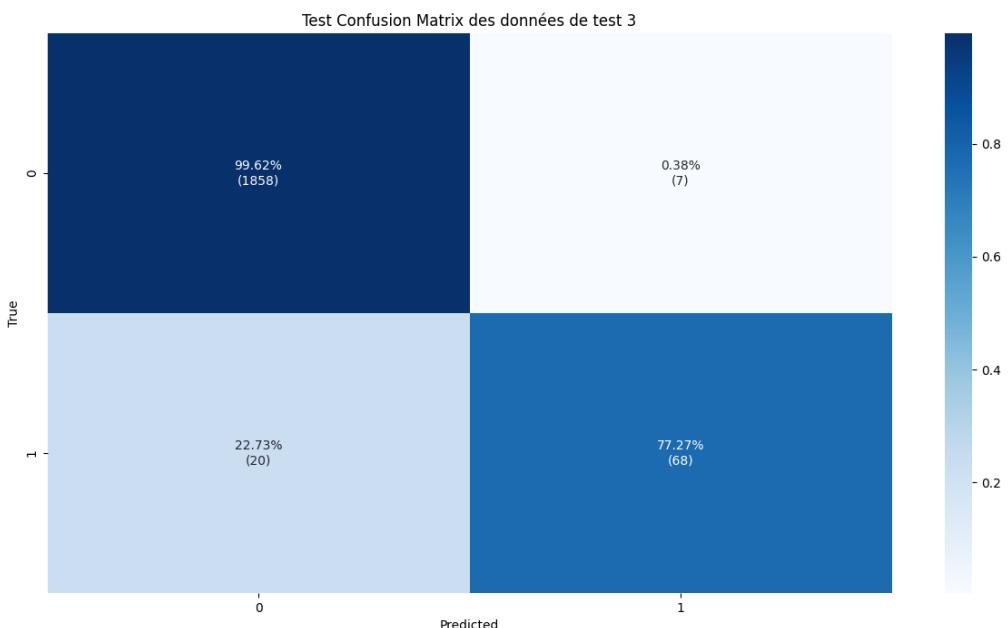


FIGURE 35 – Matrice de confusion sur les données de test pour le micro 3.

Microphone 4 (cardioïde à gauche)

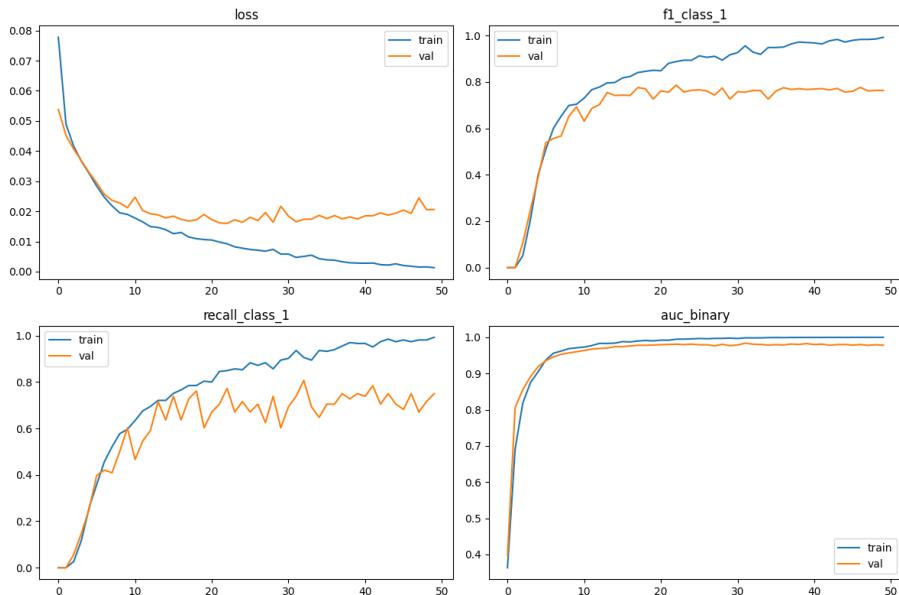


FIGURE 36 – Courbe d'entraînement pour le micro 4.

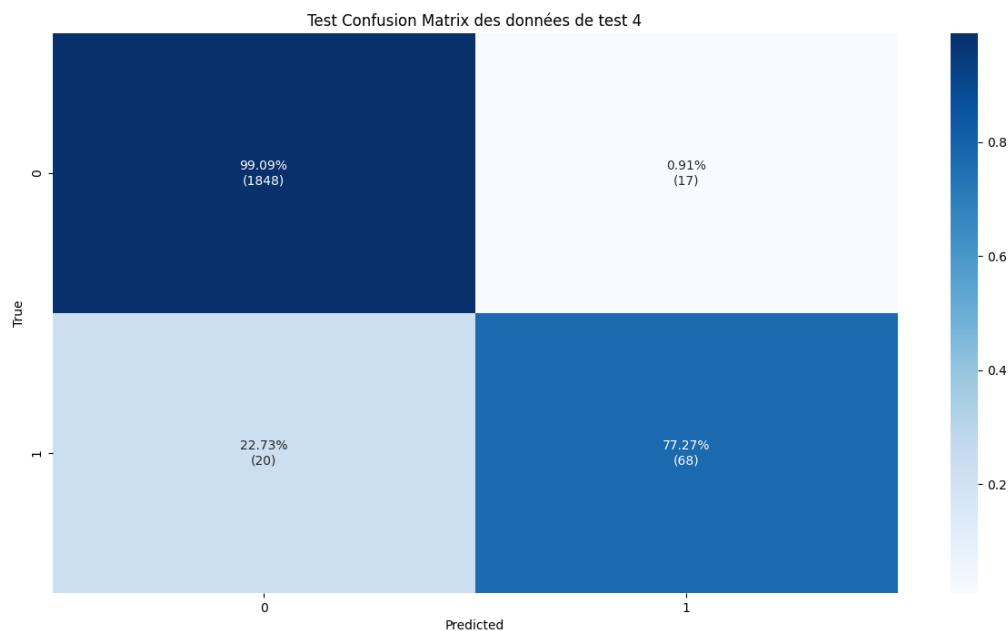


FIGURE 37 – Matrice de confusion sur les données de test pour le micro 4.

Fusion tardive des capteurs

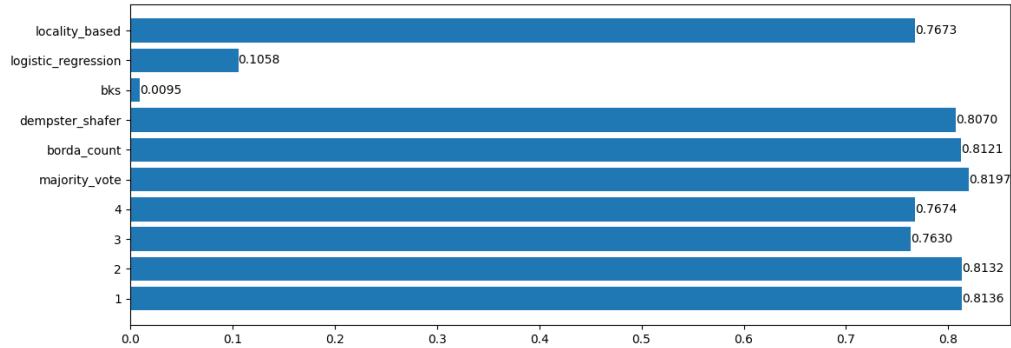


FIGURE 38 – Schéma de la fusion tardive des sorties des quatre microphones.

Les résultats obtenus montrent clairement que les performances les plus satisfaisantes proviennent des microphones 1 et 2, tandis que les microphones 3 et 4 se révèlent moins fiables et moins performants. Forts de ces observations, la prochaine étape a consisté à réaliser une fusion précoce des données issues des microphones 1 et 2, en conservant les mêmes paramètres d'entraînement que précédemment. Les résultats obtenus après cet entraînement sont présentés ci-après.

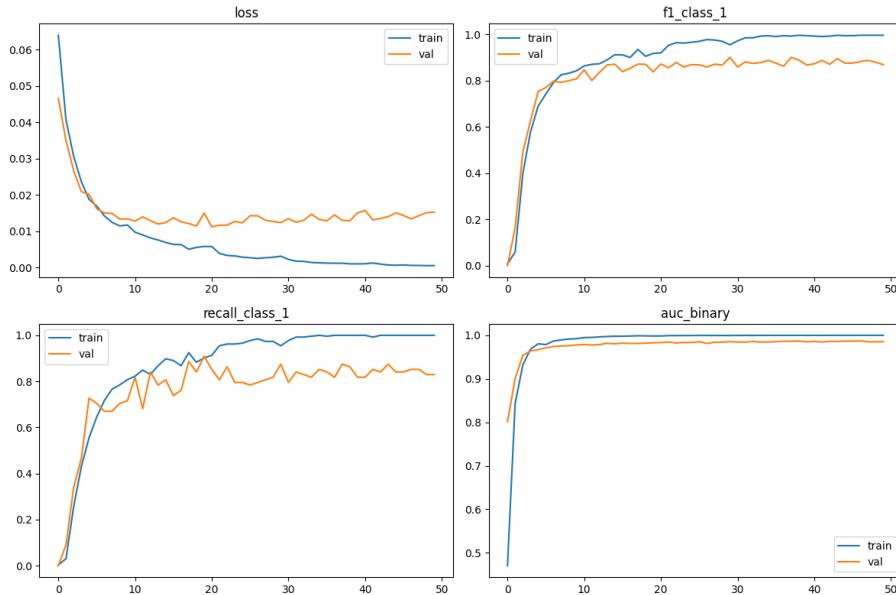


FIGURE 39 – Courbes d'entraînement (perte et précision) lors de la fusion précoce des micros 1 et 2.

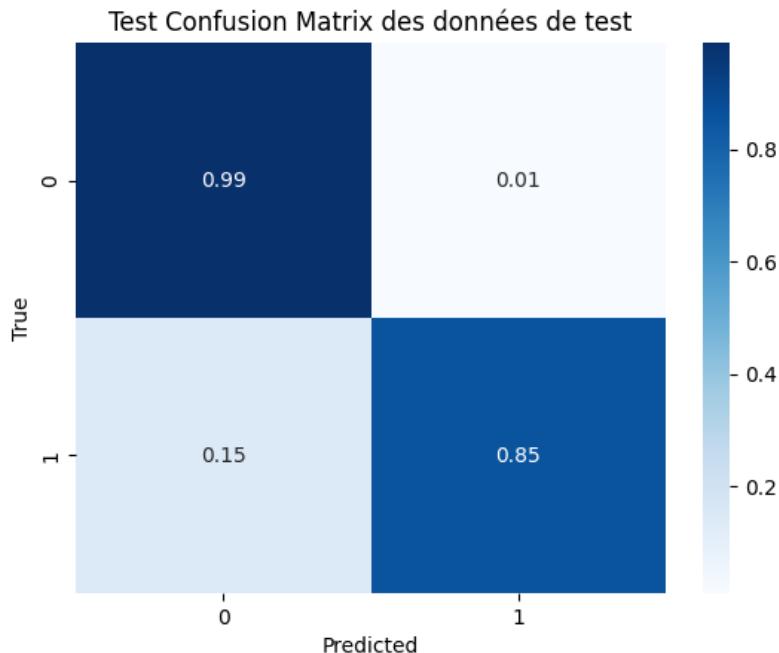


FIGURE 40 – Matrice de confusion obtenue sur les données de test après fusion précoce.

Résultats de la fusion précoce des micros 1 et 2 :

Conclusion sur la fusion précoce des données :

La fusion précoce des données issues des micros 1 et 2 a permis d'obtenir des résultats nettement supérieurs à ceux des micros individuels. En effet, cette approche réduit considérablement les faux positifs, ce qui améliore la précision globale du modèle. Cependant, on observe toujours la présence de faux négatifs.

Cela pourrait s'expliquer par le fait que les micros 1 et 2 sont situés du côté droit, ce qui peut amener le modèle à « oublier » ou mal détecter les signaux provenant du côté gauche, notamment ceux liés aux roues situées de ce côté.

Pour pallier ce problème, la prochaine étape a consisté à réaliser des fusions précoce incluant les deux micros cardioïdes (1 et 4) puis l'ensemble des quatre micros (1, 2, 3 et 4).

Les résultats obtenus pour ces différentes configurations sont comparés dans les figures suivantes, où l'on présente côté à côté les performances des architectures basées sur le micro 1 seul, la fusion des micros 1 et 2, la fusion des micros 1 et 4, ainsi que la fusion des quatre micros et enfin la fusion tardive entre ces trois catégories.

5 Approche d'apprentissage automatique pour la détection des joints

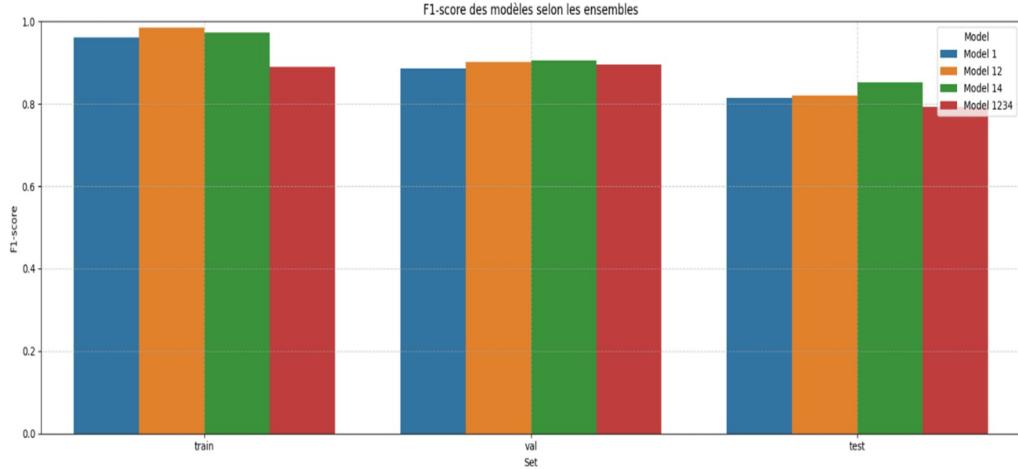


FIGURE 41 – Comparaison des F1-scores sur les données de test entre les différentes configurations : micro 1 seul, fusion micros 1 et 2, fusion micros 1 et 4, fusion des micros 1, 2, 3 et 4.

Cette figure illustre la comparaison du F1-score pour la classe 1 sur les données de test. On observe qu'il n'existe pas de différence majeure entre les méthodes comparées. Toutefois, il est évident que la fusion précoce entre les micros 1 et 2, ainsi que celle entre les micros 1 et 4, fournit les meilleurs résultats.

Cette figure présente la comparaison des matrices de confusion obtenues à partir des quatre modèles évalués. On constate globalement de bonnes performances. Cependant, les combinaisons 1-2 et 1-4 se distinguent par les meilleurs résultats.

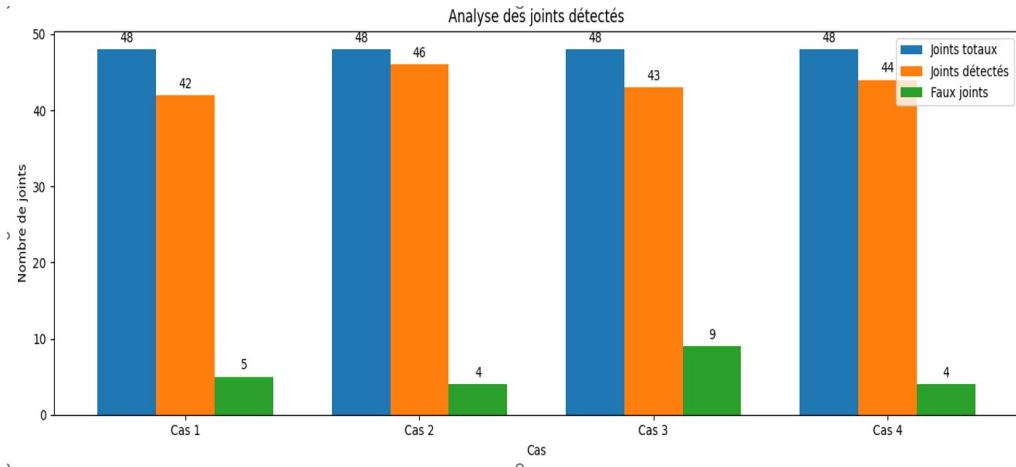


FIGURE 42 – Comparaison du comptage des vrais joints détectés pour chaque configuration : micro 1 seul, fusion micros 1 et 2, fusion micros 1 et 4, fusion des micros 1, 2, 3 et 4.

Cette figure illustre la comparaison des modèles en termes de détection correcte

5 Approche d'apprentissage automatique pour la détection des joints

des joints. La logique adoptée est la suivante : un joint est considéré comme correctement détecté si au moins une des séquences associées (correspondant à une roue) est identifiée comme positive. Dans le cas contraire, si toutes les séquences sont incorrectement détectées, il s'agit d'un faux positif.

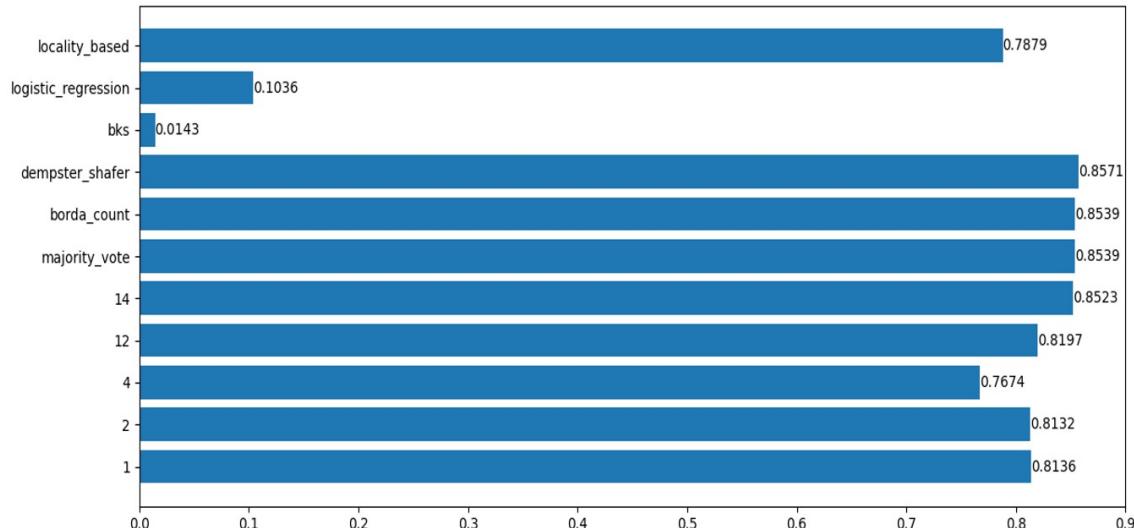


FIGURE 43 – Schéma illustrant la fusion tardive des données issues des quatre configurations micro : Micro 1 seul, Micro 2 seul, mIcro 4 seul, Fusion Micro 1 et 2, Fusion Micro 1 et 4.

On constate également qu'avec la fusion tardive, en particulier lorsqu'elle est réalisée à l'aide des méthodes de Dempster-Shafer ou de vote, les performances sont améliorées, ce qui constitue un point positif.

Bien que les résultats soient globalement satisfaisants, le modèle semble omettre certaines observations. En analysant en détail les prédictions et les visualisations, nous avons émis l'hypothèse que ces oubliés pourraient concerter principalement les roues arrière. Pour vérifier cette hypothèse, une nouvelle approche a été envisagée : utiliser la même architecture mais avec trois classes distinctes — rien, roue avant et roue arrière — afin d'évaluer si ce sont effectivement les roues arrière qui posent problème au modèle.

Résultats de l'essai avec trois classes

Les résultats présentés ci-dessous correspondent à l'expérimentation utilisant trois classes distinctes : *rien*, *roue avant* et *roue arrière*. La Figure ci-dessous illustre la matrice de confusion obtenue sur les données de test.

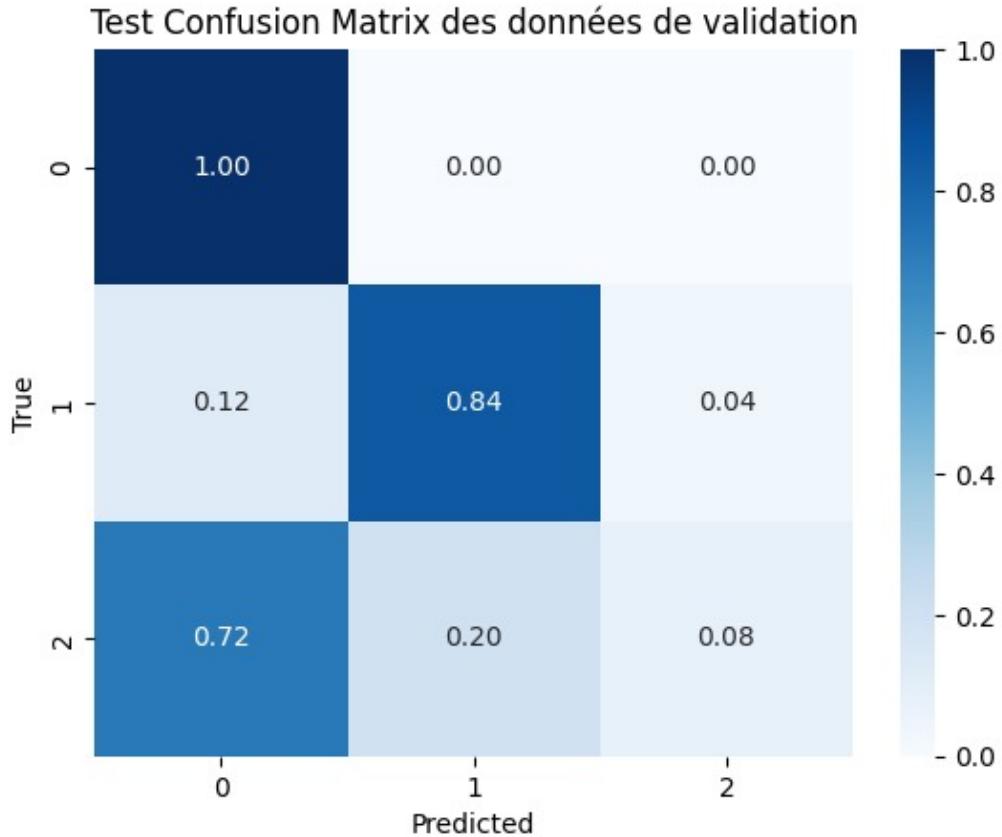


FIGURE 44 – Matrice de confusion sur les données de test.

L'analyse de la matrice de confusion révèle que la majorité des erreurs concernent la classe *roue arrière*, souvent prédictée comme *rien*. Cela suggère que le modèle rencontre des difficultés spécifiques à détecter correctement les roues arrière, ce qui pourrait expliquer certaines erreurs observées lors des essais précédents.

Pour améliorer la robustesse des modèles, l'idée a été d'intégrer une modalité supplémentaire, en l'occurrence les données issues des accéléromètres.

5.2 Modélisation basée sur les signaux accéléromètres

Dans cette partie, nous nous intéressons à l'exploitation des données issues des accéléromètres en les traitant de manière analogue à des signaux audio. L'objectif est de prédire les joints en appliquant des méthodes de traitement du signal inspirées de la reconnaissance audio, notamment à travers l'utilisation des coefficients Mels, de leurs dérivées, et en cherchant les paramètres optimaux pour la classification. Il convient toutefois de noter qu'en théorie, cette étape ne devrait pas offrir des performances aussi bonnes que pour le son, car les coefficients Mels sont une représentation spécifiquement adaptée aux signaux audio. En complément, une autre

étude a été réalisée à partir de la décomposition en ondelettes, conduite par mon encadrant, afin de comparer les approches et évaluer leur pertinence dans le cadre de séries temporelles issues de mesures vibratoires.

5.2.1 Accéléromètres utilisés et annotations

Trois accéléromètres mono-axes ont été utilisés dans cette étude :

- L'accéléromètre d'indice **1** mesure les vibrations verticales.
- L'accéléromètre d'indice **3** mesure également les vibrations verticales.
- L'accéléromètre d'indice **4** mesure quant à lui les vibrations horizontales.

Pour l'annotation des données, nous nous appuyons principalement sur les annotations déjà établies à partir des signaux audio. Cependant, afin de garantir la cohérence entre les différentes modalités, un ajustement des intervalles a été réalisé lorsque les segments temporels des accéléromètres étaient plus étendus que ceux des signaux audio.

5.2.2 Extraction des caractéristiques

Pour exploiter les signaux issus des accéléromètres, nous avons choisi de travailler avec des coefficients de type *Mel Spectrogram*. Contrairement aux coefficients cepstraux (MFCCs), cette représentation permet de conserver davantage d'informations spectrales tout en tenant compte de l'échelle perceptuelle de Mel. L'analyse du spectre fréquentiel des signaux (cf. Figure 45) montre qu'au-delà de 10 kHz, il n'existe pratiquement aucune énergie utile. En conséquence, nous avons limité la fréquence maximale à **10 kHz** ($f_{\max} = 10$ kHz) pour le calcul des coefficients Mel, ce qui permet de réduire le bruit et d'optimiser le traitement.

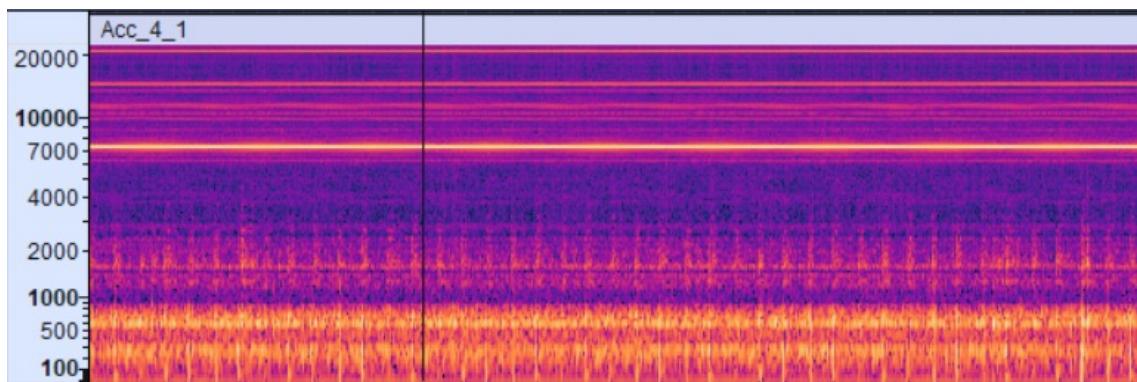


FIGURE 45 – Spectre fréquentiel d'un signal accéléromètre montrant l'absence d'énergie significative au-delà de 10 kHz.

5.2.3 Présentation de l'architecture utilisée

Afin de sélectionner une architecture adaptée, une phase préliminaire d'expérimentation a été réalisée sans recours à des méthodes d'optimisation avancées telles qu'Optuna. Nous avons volontairement limité l'entraînement à un petit nombre d'itérations (*epochs*) en utilisant un taux d'apprentissage relativement élevé afin d'identifier rapidement les configurations susceptibles de converger efficacement.

Caractéristiques en entrée. Les signaux des accéléromètres ont été représentés à l'aide de **64 coefficients Mel**, calculés sur des fenêtres de **20 ms** avec un **chevauchement de 50%**, au même format que les signaux audio pour assurer une comparabilité directe. Aux coefficients de base, nous avons ajouté leurs **dérivées premières** ainsi que l'**énergie du signal** et sa dérivée première. Le nombre de trames a été fixé à **20 trames** avec le même chevauchement, afin de produire des prédictions sur un ensemble homogène de caractéristiques.

Architecture du modèle détaillée L'architecture finale du modèle est constituée de plusieurs blocs complémentaires, conçus pour extraire efficacement les caractéristiques spatiales et temporelles des données :

- **Deux couches de convolution 2D :**

La première couche utilise 32 filtres et la seconde 64 filtres, avec un *stride* de 2 sur l'axe fréquentiel pour la deuxième couche afin de réduire la dimension tout en conservant l'information temporelle. Chaque couche de convolution est suivie d'une couche **Dropout** (typiquement 0.3-0.5) pour réduire le risque de surapprentissage en désactivant aléatoirement une fraction des neurones pendant l'entraînement. Ces couches permettent au réseau d'apprendre des caractéristiques locales pertinentes dans les signaux.

- **Deux couches LSTM bidirectionnelles :**

La première LSTM bidirectionnelle contient 64 unités et la seconde 32 unités, ce qui permet de capturer les dépendances temporelles dans les deux directions. Chaque LSTM est suivie d'une couche **Dropout** pour régulariser le modèle et éviter le surapprentissage, en particulier sur les séquences où les patterns peuvent être répétitifs.

- **Couche Dense finale :**

La sortie des LSTM est transmise à une couche fully-connected (*Dense*) de deux neurones, correspondant aux deux classes à prédire (présence ou absence de joint). Une activation **softmax** est utilisée pour produire des probabilités normalisées pour chaque classe, permettant au modèle de fournir une prédiction interprétable comme une distribution de probabilité.

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 20, 130)	0
reshape (Reshape)	(None, 20, 130, 1)	0
conv2D_0 (Conv2D)	(None, 20, 130, 32)	512
leaky_re_lu (LeakyReLU)	(None, 20, 130, 32)	0
dropout (Dropout)	(None, 20, 130, 32)	0
conv2D_1 (Conv2D)	(None, 20, 65, 64)	18,496
leaky_re_lu_1 (LeakyReLU)	(None, 20, 65, 64)	0
dropout_1 (Dropout)	(None, 20, 65, 64)	0
reshape_1 (Reshape)	(None, 20, 4160)	0
bidirectional (Bidirectional)	(None, 20, 64)	1,073,408
leaky_re_lu_2 (LeakyReLU)	(None, 20, 64)	0
dropout_2 (Dropout)	(None, 20, 64)	0
bidirectional_1 (Bidirectional)	(None, 32)	10,368
leaky_re_lu_3 (LeakyReLU)	(None, 32)	0
dropout_3 (Dropout)	(None, 32)	0
dense (Dense)	(None, 2)	66

FIGURE 46 – Résumé de l'architecture finale utilisée pour la classification des signaux accéléromètres.

5.2.4 Résultats par accéléromètre individuel

Afin d'évaluer les performances du modèle, nous avons analysé séparément les signaux provenant de chaque accéléromètre. Les résultats de l'entraînement seront présentés sous la forme d'une **matrice de confusion** calculée sur le *set* de test, contenant 88 séquences de joints à détecter. De plus, nous présenterons la **courbe ROC** pour les trois ensembles (train, validation et test), permettant d'observer l'aire sous la courbe (AUC) correspondante dans chaque *set*, afin de mesurer la qualité globale de la classification.

Accéléromètre 1

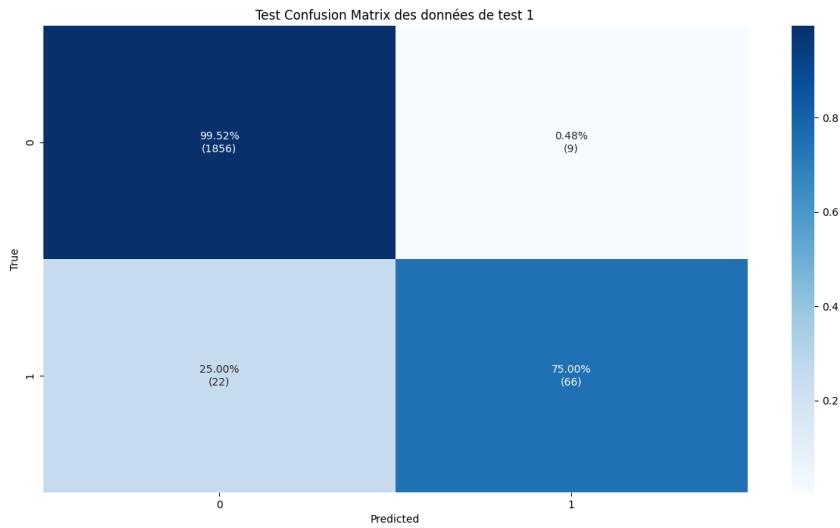


FIGURE 47 – Résultats pour l'accéléromètre 1.

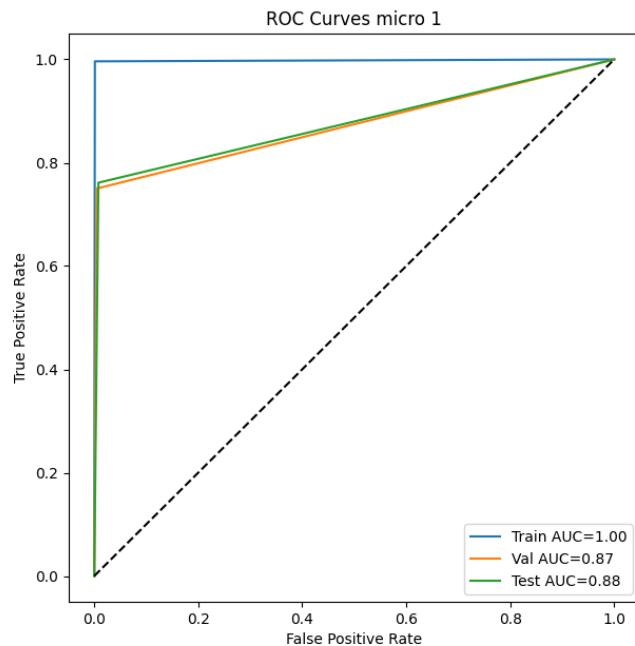


FIGURE 48 – Résultats pour l'accéléromètre 1.

Accéléromètre 3

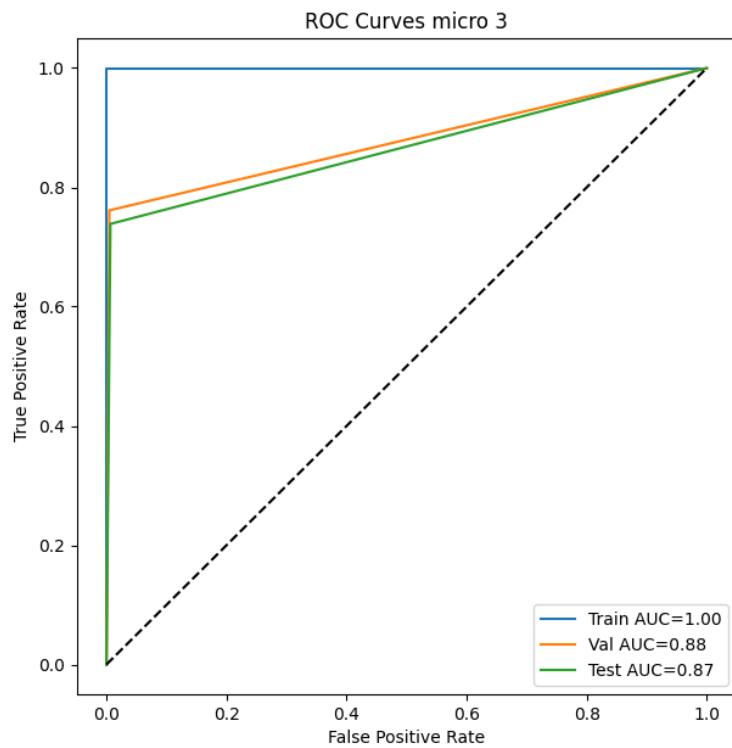
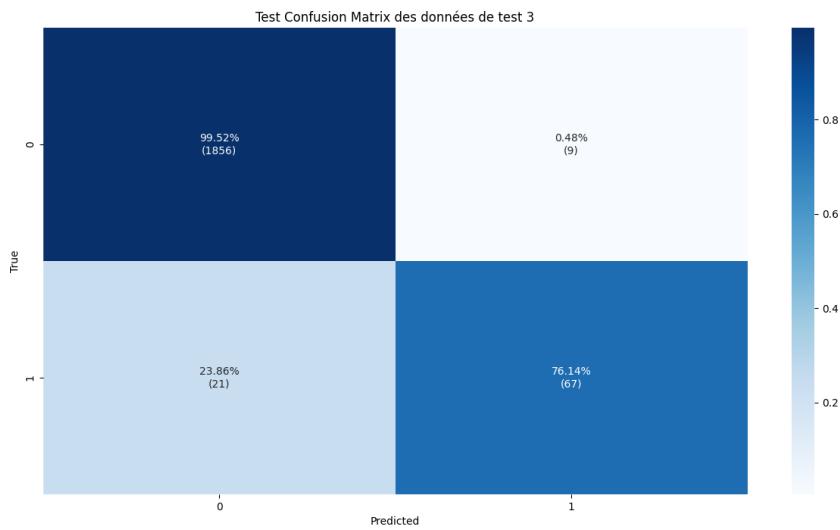


FIGURE 49 – Résultats pour l'accéléromètre 3

Accéléromètre 4

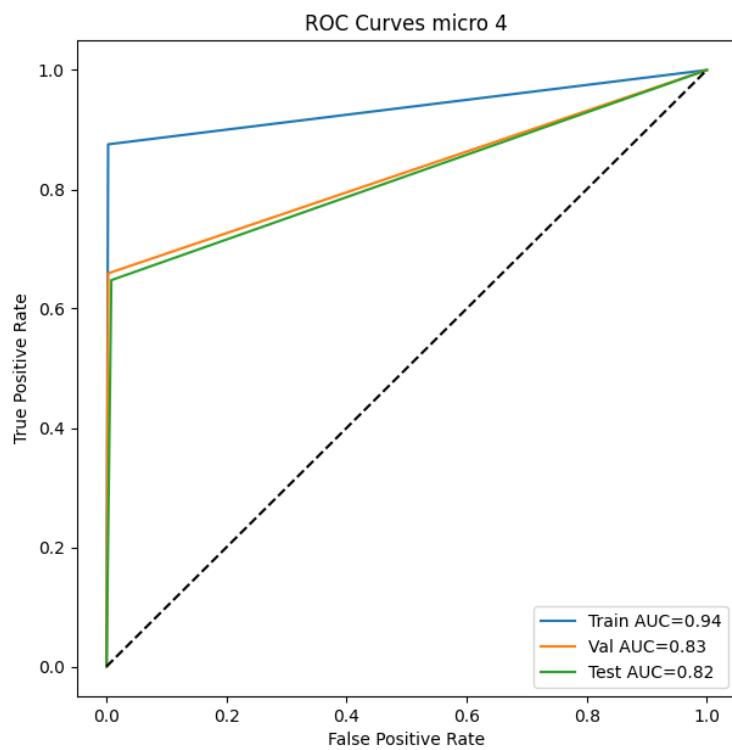
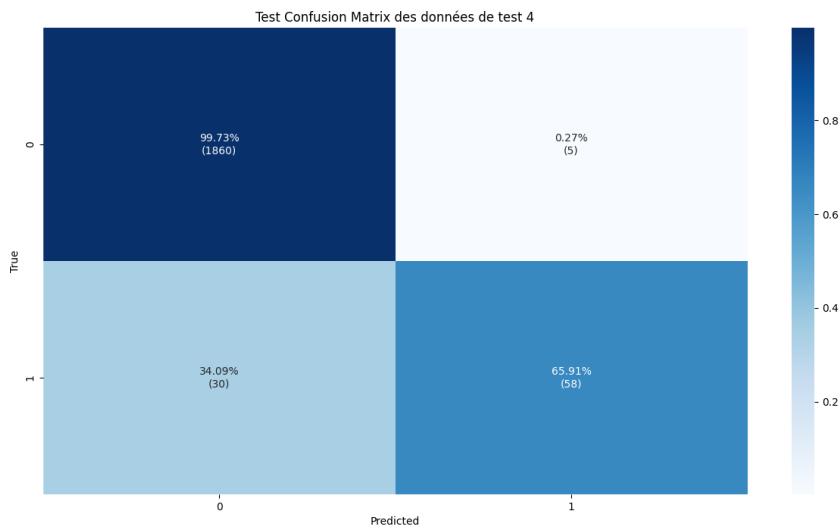


FIGURE 50 – Résultats pour l'accéléromètre 4

5.2.5 Fusion Précoce

L'étude précédente a montré les performances des modèles entraînés à partir de chaque accéléromètre pris individuellement. Afin d'améliorer la robustesse et de tirer parti de la complémentarité des signaux, nous avons ensuite étudié la **fusion précoce de ces plusieurs capteurs**.

Trois configurations ont été considérées :

- accéléromètres 1 et 3 (*deux signaux verticaux*) ;
- accéléromètres 3 et 4 (*un signal vertical et un signal horizontal*) ;
- accéléromètres 1, 3 et 4 (*fusion complète*).

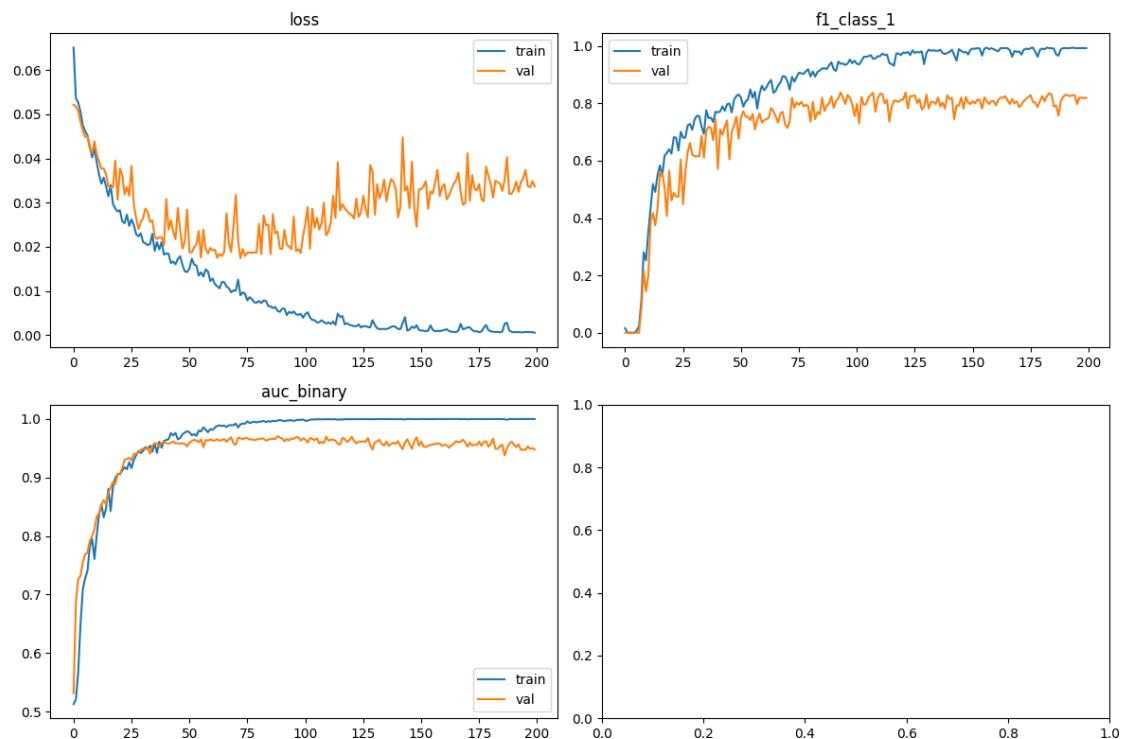


FIGURE 51 – Courbes d'apprentissage pour la fusion des accéléromètres 1 et 3 : perte (loss/val_loss), F1-score de la classe *joint*, et AUC.

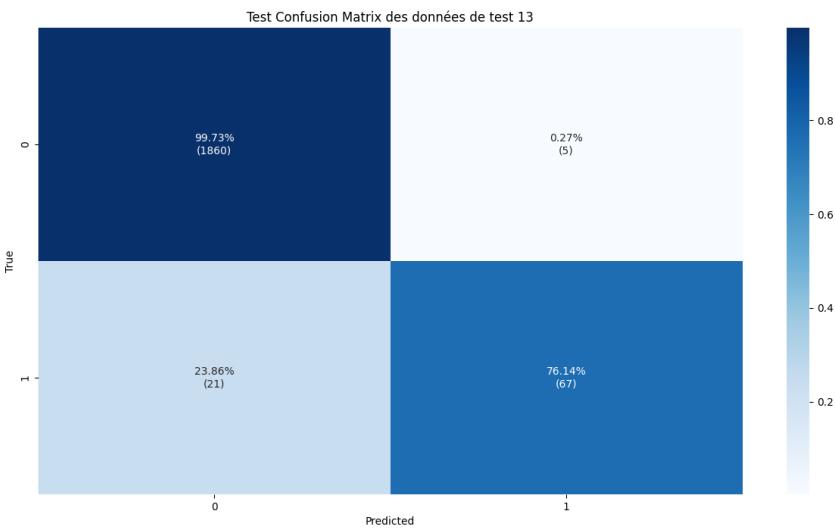


FIGURE 52 – Matrice de confusion sur l'ensemble de test pour la fusion des accéléromètres 1 et 3.

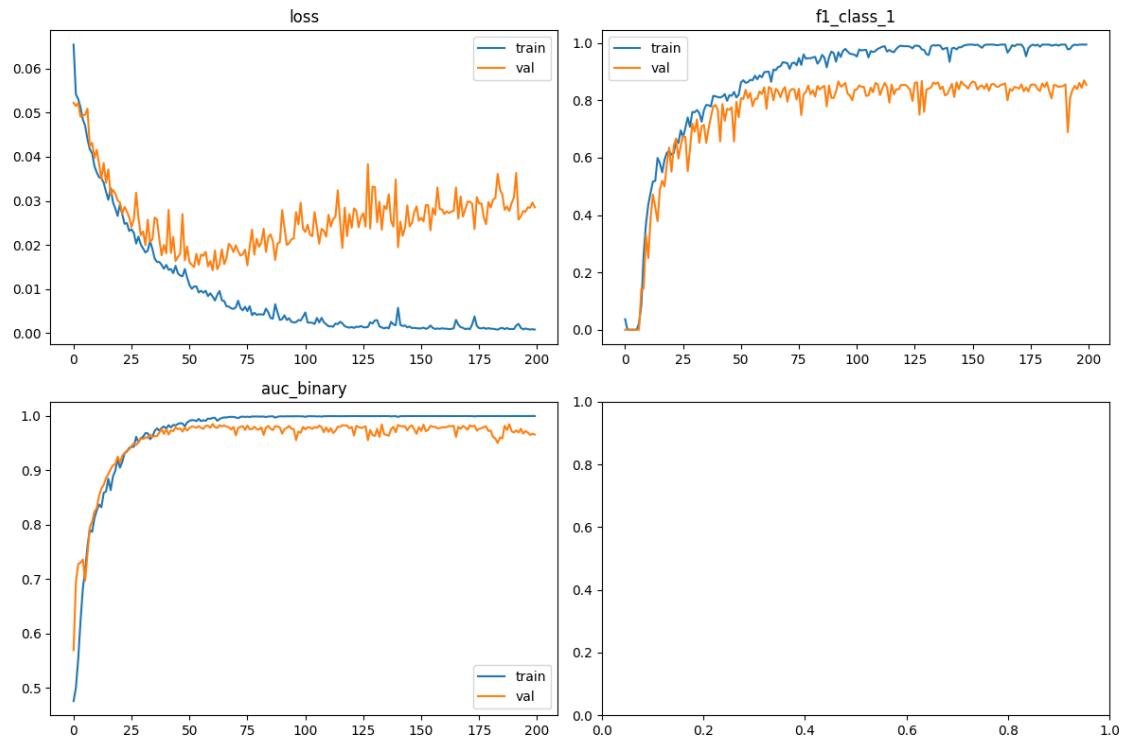


FIGURE 53 – Courbes d'apprentissage pour la fusion des accéléromètres 3 et 4.

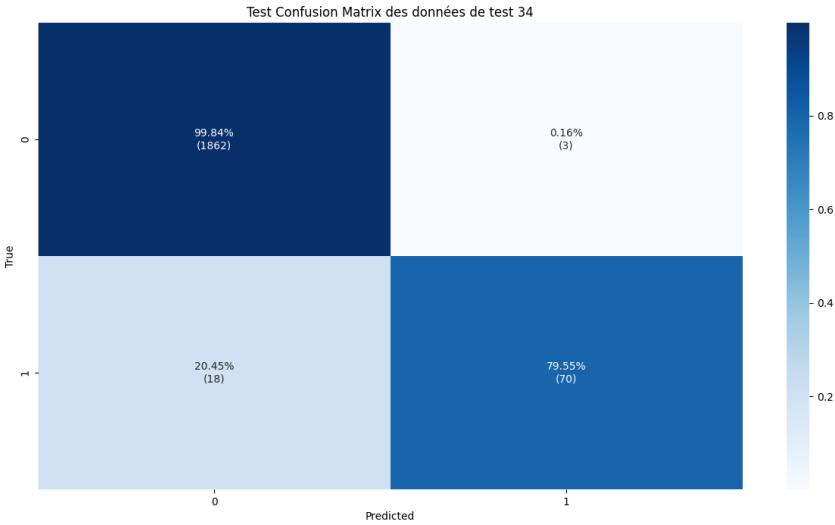


FIGURE 54 – Matrice de confusion sur l'ensemble de test pour la fusion des accéléromètres 1 et 4.

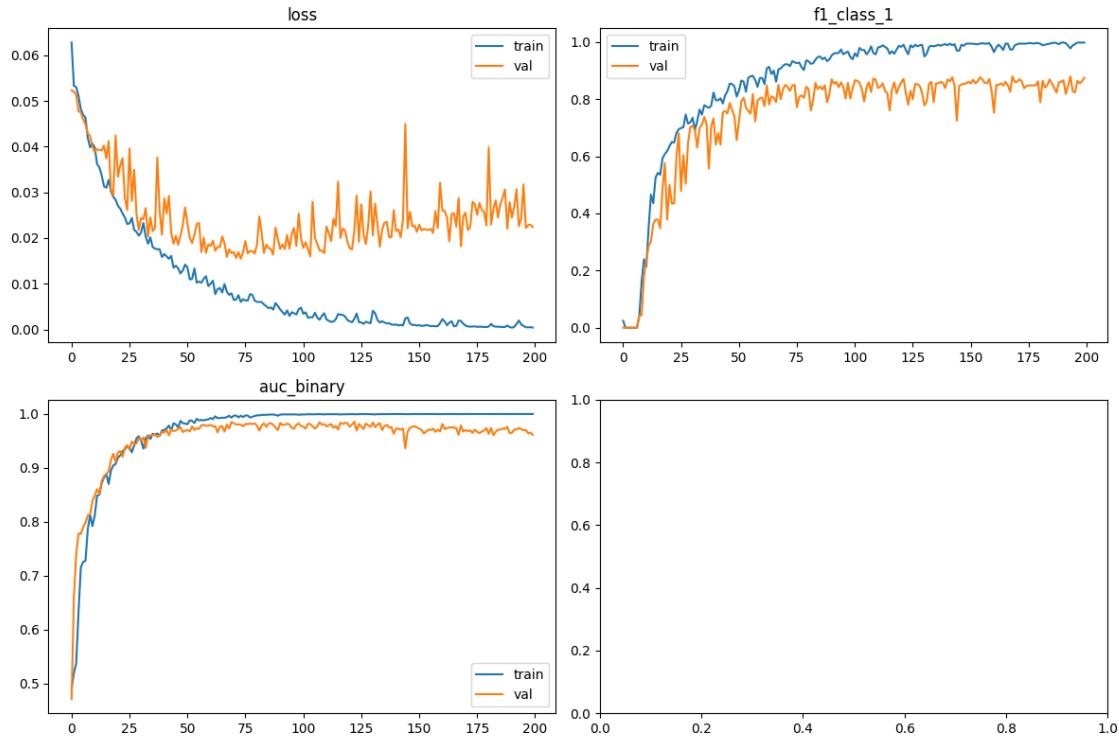


FIGURE 55 – Courbes d'apprentissage pour la fusion des accéléromètres 1, 3 et 4.

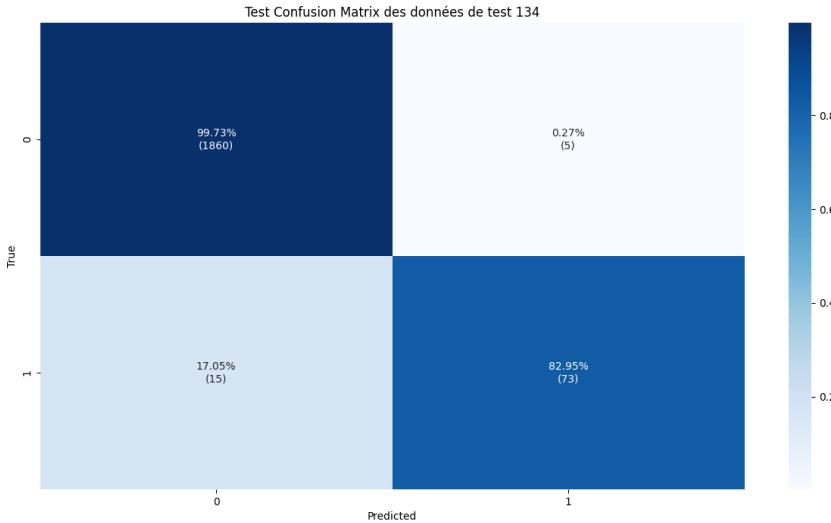


FIGURE 56 – Matrice de confusion sur l'ensemble de test pour la fusion des accéléromètres 1, 3 et 4.

On observe un fort surapprentissage dans toutes les courbes de loss, ce qui peut s'expliquer par l'inadéquation des coefficients Mel pour le traitement des signaux de vibrations.

5.2.6 Fusion tardive des accéléromètres

Dans ce cas, les prédictions issues des modèles entraînés séparément sur les accéléromètres 1, 3 et 4 ont été fusionnées (par exemple via une moyenne pondérée des probabilités).

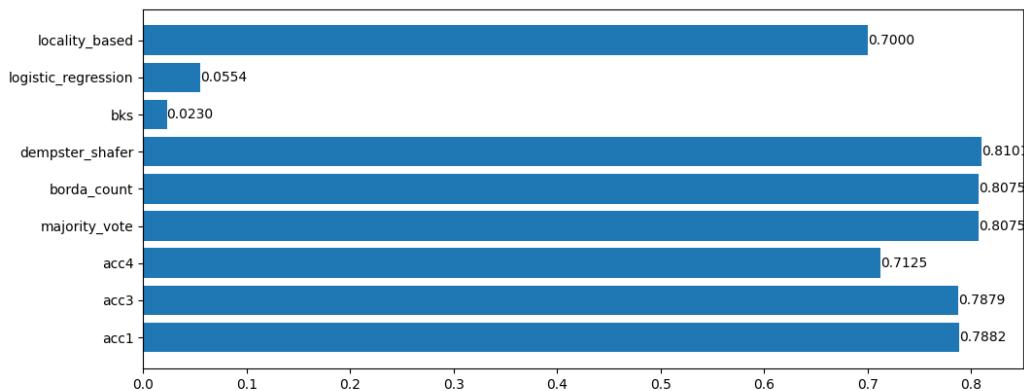


FIGURE 57 – F1 score sur les données de test de la classe joint.

Fusion tardive entre modèles de fusion précoce. Dans un second scénario, nous avons appliqué la fusion tardive entre les modèles entraînés sur les combinaisons précoce (3 et 4) et (1, 3 et 4). Cette stratégie vise à comparer la complémentarité entre modèles précoce et individuels.

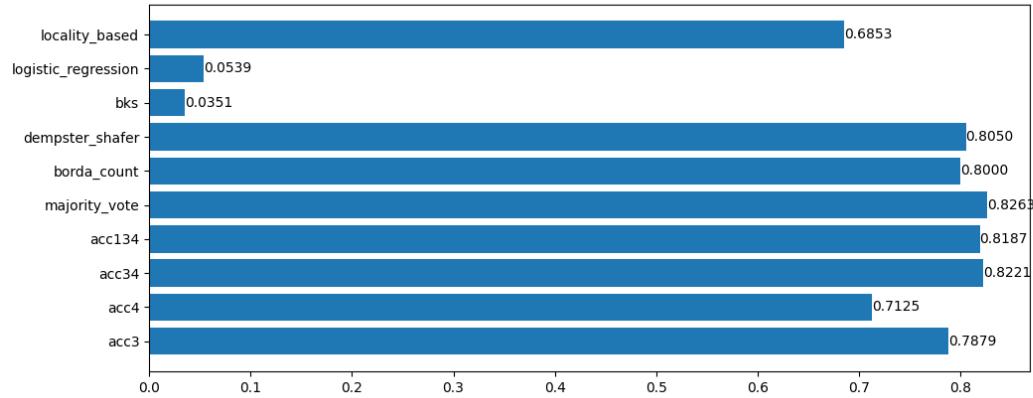


FIGURE 58 – F1 score sur les données de test de la classe joint.

D'après l'ensemble des résultats, on peut conclure que, pris individuellement, l'accéléromètre 3 fournit les meilleures performances sur les données de test, avec une bonne convergence des courbes d'apprentissage. En revanche, l'accéléromètre 1, bien qu'il capte les mêmes vibrations verticales, présente des résultats moins satisfaisants, et l'accéléromètre horizontal ne montre pas non plus de bonnes performances. On observe toutefois que la fusion tardive permet d'améliorer légèrement les scores, mais ce sont les approches de fusion précoce des trois capteurs qui donnent les meilleurs résultats, offrant un compromis solide entre stabilité et performance.

Dans la suite, nous allons tester la capacité de généralisation du modèle en réalisant des prédictions sur le jeu de données “aller”, caractérisé par une vitesse correspondant au double de celle utilisée dans le fichier d'entraînement.

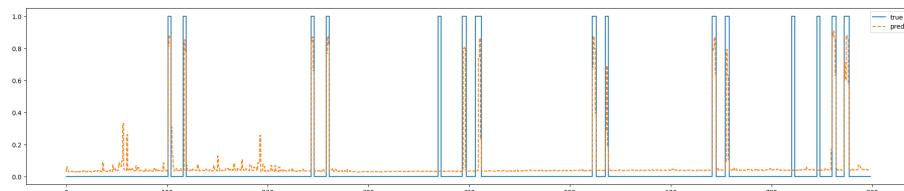


FIGURE 59 – Probabilités prédites par le modèle audio en fonction des vrais labels sur le set *aller*.

5 Approche d'apprentissage automatique pour la détection des joints

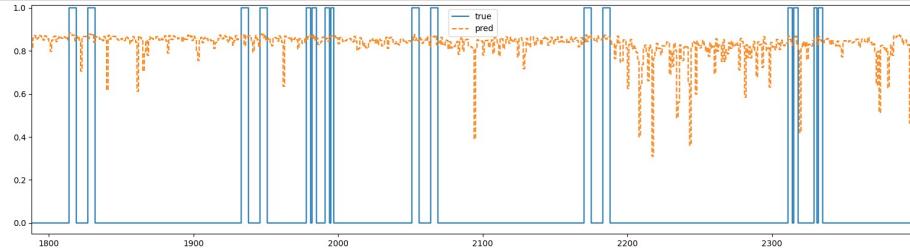


FIGURE 60 – Probabilités prédictes par le modèle basé sur les accéléromètres en fonction des vrais labels sur le set *aller*.

Les résultats montrent une différence marquée entre les deux approches. Pour les signaux audios, le modèle parvient à détecter correctement les motifs caractéristiques, même lorsque la vitesse est modifiée (réduite de moitié par rapport à l'entraînement). En revanche, les modèles basés sur les accéléromètres se révèlent beaucoup moins robustes : ils peinent à s'adapter à ce changement important de vitesse et n'arrivent pas à conserver des performances satisfaisantes. Cela suggère que les représentations extraites des audios sont plus invariantes aux variations de vitesse que celles issues des accéléromètres.

Alors , La prochaine étape consiste à explorer la fusion entre les deux modalités (audios et accéléromètres), afin d'évaluer si l'apport des accéléromètres peut contribuer à améliorer les performances obtenues avec les audios seuls.

5.3 Fusion des modalités audio et accéléromètres

5.3.1 Présentation des caractéristiques utilisées

Dans cette étude, nous avons extrait des représentations temps-fréquence adaptées à chaque modalité. Pour les signaux audio, une fréquence maximale de 22 kHz a été considérée, et les caractéristiques extraites sont constituées de 64 coefficients de Mel, accompagnés de leurs dérivées premières, ainsi que de l'énergie et de sa dérivée. De manière similaire, pour les signaux issus des accéléromètres, nous avons fixé une fréquence maximale à 10 kHz et extrait également 64 coefficients de Mel, leurs dérivées premières, l'énergie et sa dérivée.

Étant donné le nombre élevé de caractéristiques générées, une réduction de dimension par analyse en composantes principales (PCA) a d'abord été envisagée. Toutefois, les résultats obtenus sont restés insatisfaisants. Une hypothèse avancée est que la PCA tend à altérer l'information temporelle, ce qui nuit particulièrement aux modèles séquentiels tels que les LSTMs. Pour pallier ce problème, nous avons testé une approche consistant à construire des vecteurs enrichis intégrant chaque observation ainsi que ses voisins temporels avant d'appliquer la PCA. Cette méthode a montré de meilleures performances, mais au prix d'un temps d'entraînement significativement plus élevé.

5.3.2 Présentation de l'architecture utilisée

La sélection de l'architecture optimale (nombre de couches, taille des couches cachées, taille de batch, taux d'apprentissage, etc.) a été réalisée de manière manuelle. Pour cela, nous avons exploité l'ensemble des capteurs disponibles en choisissant initialement un *batch size* relativement grand ainsi qu'un taux d'apprentissage élevé. L'objectif de cette configuration était de tester rapidement la capacité du modèle à converger, indépendamment de la stabilité finale de l'apprentissage.

À l'issue de ces expérimentations exploratoires, l'architecture retenue est celle qui montrait la meilleure stabilité et une convergence satisfaisante dans ces conditions accélérées. Cette architecture est représentée ci-dessous à travers un extrait de la fonction `model.summary()` :

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 20, 910)	0
reshape (Reshape)	(None, 20, 910, 1)	0
conv2D_0 (Conv2D)	(None, 20, 455, 64)	1,024
leaky_re_lu (LeakyReLU)	(None, 20, 455, 64)	0
dropout (Dropout)	(None, 20, 455, 64)	0
conv2D_1 (Conv2D)	(None, 20, 228, 32)	18,464
leaky_re_lu_1 (LeakyReLU)	(None, 20, 228, 32)	0
dropout_1 (Dropout)	(None, 20, 228, 32)	0
conv2D_2 (Conv2D)	(None, 20, 114, 16)	4,624
leaky_re_lu_2 (LeakyReLU)	(None, 20, 114, 16)	0
dropout_2 (Dropout)	(None, 20, 114, 16)	0
reshape_1 (Reshape)	(None, 20, 1824)	0
bidirectional (Bidirectional)	(None, 20, 256)	1,999,872
leaky_re_lu_3 (LeakyReLU)	(None, 20, 256)	0
dropout_3 (Dropout)	(None, 20, 256)	0
bidirectional_1 (Bidirectional)	(None, 128)	164,352
leaky_re_lu_4 (LeakyReLU)	(None, 128)	0
dropout_4 (Dropout)	(None, 128)	0
dense (Dense)	(None, 2)	258

FIGURE 61 – Résumé de l'architecture finale retenue.

5.3.3 Résultats des fusions multimodales audio–accéléromètres

Dans cette section, nous comparons systématiquement chaque fusion testée à la configuration de référence, à savoir la fusion précoce entre les microphones cardioïdes 1 et 2. Toutes les figures présentées correspondent à une comparaison directe entre l'essai considéré et cette configuration de référence.

Fusion Mic1 + Acc3. La première expérimentation consiste à combiner le microphone cardioïde 1 avec l'accéléromètre vertical 3. La figure suivante illustre la comparaison avec la configuration de référence **Mic1 + Mic2**.

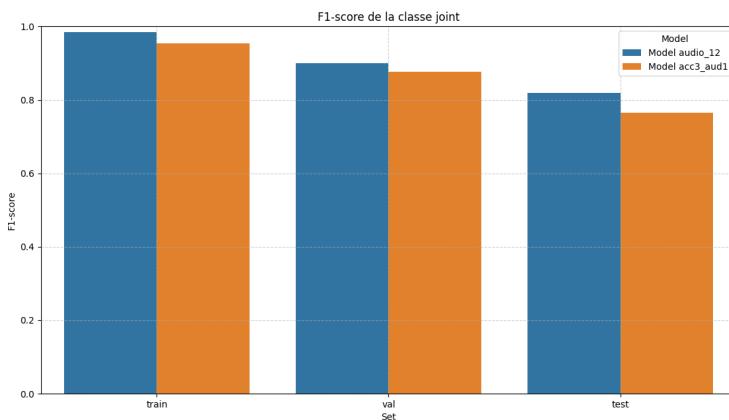


FIGURE 62 – Comparaison entre la fusion **Mic1 + Acc3** et la référence **Mic1 + Mic2**.

On constate que les performances de la fusion **Mic1 + Acc3** sont proches de celles de la configuration de référence, mais que la fusion **Mic1 + Mic2** reste globalement meilleure.

Fusion Mic1 + Mic2 + Acc3. Nous combinons ensuite les deux meilleurs microphones (Mic1 et Mic2) avec l'accéléromètre vertical 3. La comparaison avec la référence est illustrée dans la figure suivante.

5 Approche d'apprentissage automatique pour la détection des joints

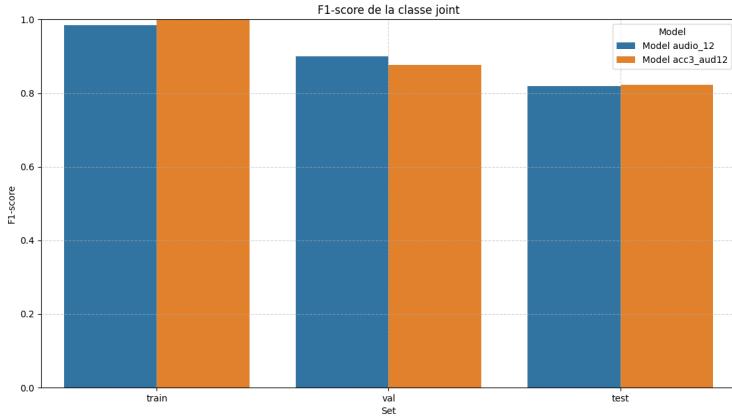


FIGURE 63 – Comparaison entre la fusion **Mic1 + Mic2 + Acc3** et la référence **Mic1 + Mic2**.

L'analyse montre que la fusion **Mic1 + Mic2 + Acc3** améliore légèrement les performances par rapport à la configuration **Mic1 + Mic2** sur le jeu de test, ce qui constitue un bon signe en termes de généralisation. En revanche, sur les ensembles d'entraînement et de validation, les résultats restent quasiment identiques. On peut donc conclure que l'accéléromètre vertical 3 apporte une information complémentaire aux deux microphones, utile surtout pour le test.

Après cela, nous avons décidé d'ajouter à ces trois capteurs l'accéléromètre 4 (horizontal), qui apporte nécessairement de nouvelles informations, afin d'évaluer s'il permet d'améliorer davantage les résultats.

Fusion Mic1 + Mic2 + Acc3 + Acc4. :

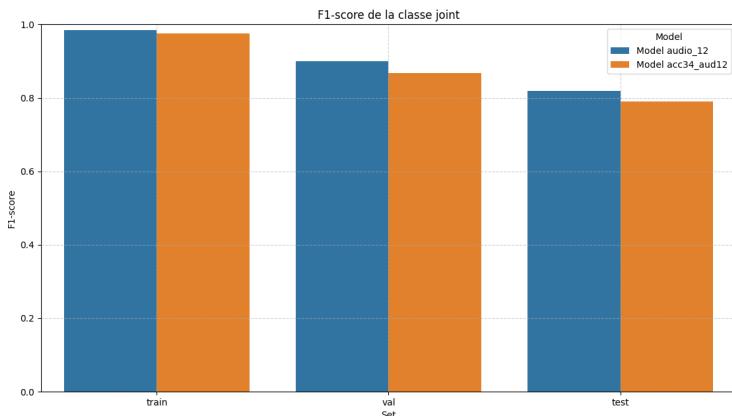


FIGURE 64 – Comparaison entre la fusion **Mic1 + Mic2 + Acc3 + Acc4** et la référence **Mic1 + Mic2**.

Les résultats suggèrent que l'ajout de l'accéléromètre horizontal n'améliore pas

5 Approche d'apprentissage automatique pour la détection des joints

significativement les performances. Ainsi, les microphones cardioïdes 1 et 2 constituent déjà une source d'information suffisante pour obtenir de bons résultats.

Enfin, nous avons tenté de fusionner l'ensemble des sept capteurs, et nous présentons ci-dessous les résultats obtenus en comparaison avec la configuration de référence utilisant uniquement les microphones 1 et 2.

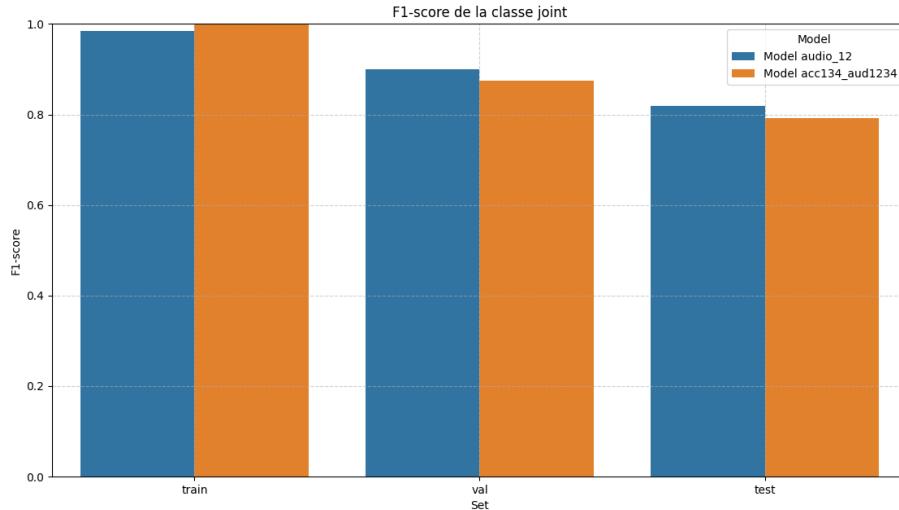


FIGURE 65 – Comparaison des performances entre la fusion de tous les capteurs (7 capteurs) et la configuration de référence (Mic1 + Mic2).

En conclusion, ces résultats montrent que l'ajout d'un grand nombre de capteurs ne garantit pas nécessairement une amélioration des performances. Au contraire, il est préférable de sélectionner uniquement les capteurs les plus pertinents. D'après nos expérimentations, la meilleure configuration est obtenue avec la combinaison des microphones 1 et 2 et de l'accéléromètre 3, qui fournit un compromis efficace entre performance et complexité.

6 Conclusion

En conclusion, les résultats indiquent que la fusion, qu'elle soit précoce ou tardive, améliore systématiquement les performances par rapport aux modèles individuels.

L'étude de l'application des coefficients Mel-spectrogramme aux signaux d'accéléromètres s'est révélée peu pertinente, car toute l'énergie des vibrations est concentrée dans les basses fréquences. La majorité des coefficients représente les hautes fréquences où il n'y a pratiquement pas d'information utile, ce qui limite l'efficacité de cette approche. Pour pallier ce problème, une autre méthode basée sur l'analyse par ondelettes, développée par mon encadrant, a été explorée.

D'autres essais ont été menés avec des réseaux TCN (Temporal Convolutional Network) prenant le signal brut en entrée, mais les résultats obtenus n'étaient pas satisfaisants.

Pour les signaux audio, les modèles ont montré de bonnes performances sur le fichier de test « aller », mais sont sensibles aux changements de conditions environnementales ou de vitesse, car les données d'entraînement présentaient une vitesse très stable.

Pour améliorer la généralisation, l'idée c'est d'augmenter le jeu de données en ajoutant du bruit sur les données brutes et testé la robustesse des modèles en entraînant sur un microphone et en testant sur un autre, et Une autre approche consiste à combiner les données de plusieurs microphones comme nouvelles observations et à entraîner un modèle unique sur l'ensemble des capteurs, en réalisant la fusion le long de l'axe des caractéristiques (axe 0).

Concernant la détection des roues arrière, certaines peuvent ne pas être entendues, ce qui entraîne parfois des omissions par les modèles. Cependant, en considérant qu'un joint est correctement détecté si au moins une roue est détectée, les erreurs liées aux roues arrière ne posent pas de problème majeur.

Enfin, l'intégration des images linéaires comme troisième modalité a été explorée. Étant donné qu'un joint apparaît une seule fois sur l'image (contrairement aux quatre passages audibles dans les signaux audio), une fusion précoce entre les trois modalités n'était pas réalisable. Toutefois, une fusion tardive après entraînement d'un modèle sur les images linéaires reste possible et constitue une piste intéressante pour des travaux futurs.

Références

- [1] (PDF) The new UIC catalogue of rail defects. *ResearchGate*, August 2025.
- [2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.
- [3] Anmol Jain, Aishwary Kumar, and Seba Susan. Evaluating Deep Neural Network Ensembles by Majority Voting Cum Meta-Learning Scheme. In V. Sivakumar Reddy, V. Kamakshi Prasad, Jiacun Wang, and K. T. V. Reddy, editors, *Soft Computing and Signal Processing*, volume 1340, pages 29–37. Springer Singapore, Singapore, 2022. Series Title : Advances in Intelligent Systems and Computing.
- [4] Jürg Kohlas and Paul-André Monney. *A Mathematical Theory of Hints*, volume 425 of *Lecture Notes in Economics and Mathematical Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.
- [5] Zuozhu Liu, Wenyu Zhang, Shaowei Lin, and Tony Q.S. Quek. Heterogeneous Sensor Data Fusion By Deep Multimodal Encoding. *IEEE Journal of Selected Topics in Signal Processing*, 11(3) :479–491, April 2017.
- [6] Jun-Ki Min and Sung-Bae Cho. Multiple Classifier Fusion Using k-Nearest Localized Templates. In Hujun Yin, Peter Tino, Emilio Corchado, Will Byrne, and Xin Yao, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, volume 4881, pages 447–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. Series Title : Lecture Notes in Computer Science.
- [7] Dymitr Ruta and Bogdan Gabrys. An Overview of Classifier Fusion Methods.
- [8] Chunli Sun and Feng Zhao. Multi-level feature fusion network for neuronal morphology classification. *Frontiers in Neuroscience*, 18 :1465642, October 2024.
- [9] Huijuan Zhao, Jusheng Mi, and Meishe Liang. A multi-granularity information fusion method based on logistic regression model and Dempster-Shafer evidence theory and its application. *International Journal of Machine Learning and Cybernetics*, 13(10) :3131–3142, October 2022.