
Rapport de Stage de Fin d'Études



INSTITUT ////////////////
DES SCIENCES ETIENNE
DU MOUVEMENT JULES
//////////////// MAREY

Ce stage s'est déroulé au sein de l'Institut des Sciences du Mouvement (ISM), un laboratoire de recherche situé sur le campus de Luminy, dans le 9e arrondissement de Marseille (13009).

La soutenance de ce rapport de stage a lieu en septembre 2025 et est présentée par Madou Fall, à l'issue d'un parcours professionnel, technique et scientifiquement enrichissant.

Sujet : LinLED, Détection de l'intention humaine par gestes simples

Interface réactive combinant apprentissage automatique et capteurs linéaires bio-inspirés

Encadré par :

DOMINIQUE MARTINEZ dominique.martinez@univ-amu.fr

STÉPHANE VIOLLET stephane.viollet@univ-amu.fr

JOCELYN MONNOYER jocelyn.monnoyer@stellantis.com

Réalisé par :

FALL MADOU madou.fall@etu.univ-amu.fr

Remerciements

Avant tout, je tiens à exprimer ma profonde gratitude envers mes superviseurs. Stéphane Viollet, instigateur du projet LinLED et auteur de la demande de brevet y afférente, m'a transmis son enthousiasme pour l'innovation et sa perspective. Sa précision scientifique, associée à son ardent intérêt pour le projet, a été une source d'inspiration majeure et a guidé toutes mes prises de décisions lors de ce stage. Dominique Martinez, spécialiste en intelligence artificielle et en classification dynamique des signaux, m'a apporté son soutien avec persévérance et expertise lors de l'analyse et de la compréhension des signaux. Ses recommandations avisées et perspicaces m'ont aidé à progresser et à surmonter les obstacles techniques auxquels j'étais confronté. Pour finir, Jocelyn Monnoyer, qui travaille comme ingénieur chez Stellantis, m'a assisté dans l'orientation de LinLED vers une interface utilisateur économiquement avantageuse. Sa perspective pratique et ses conseils judicieux ont été cruciaux pour créer un pont entre la théorie et les mises en œuvre réelles dans le domaine de l'automobile.

Je voudrais aussi exprimer ma profonde gratitude à toute l'équipe du projet LinLED, qui a considérablement enrichi et dynamisé ma tâche quotidienne. Mouhamed Elkairouh, un ingénieur spécialisé en intelligence artificielle embarquée, a partagé son savoir-faire concernant les algorithmes d'apprentissage automatique. Il m'a aidé en particulier dans l'implémentation des premières classifications utilisant Random Forest. Sa présence constante et sa patience m'ont été indispensables pour saisir et mettre en pratique ces notions complexes. William Wilmot, un ingénieur polyvalent et passionné, m'a guidé sur la manière d'utiliser LinLED. Sa bonne humeur, son dynamisme et son sens de l'enseignement ont rendu chaque phase du projet plus évidente et plaisante. Jean-Marc Ingargiola, spécialiste en électronique, a conçu les cartes électroniques et pris en charge le traitement des données. Sa guidance détaillée, sa présence constante et sa largesse dans la transmission de ses connaissances ont été d'une grande utilité tout au long du projet. En outre, Julien Serres, Professeur des Universités, a profondément nourri ma pensée scientifique à travers ses séminaires captivants et nos échanges enrichissants sur les diverses méthodologies de recherche.

J'éprouve aussi une profonde gratitude envers toute l'équipe de recherche pour leur accueil chaleureux et leur esprit de collaboration. Les discussions scientifiques, les conversations informelles et l'atmosphère bienveillante ont été une véritable source d'encouragement et d'inspiration. Être dans un cadre aussi stimulant m'a donné l'opportunité non seulement d'améliorer mes aptitudes techniques, mais également de grandir personnellement et d'acquérir des compétences interpersonnelles cruciales.

Je tiens aussi à exprimer ma gratitude envers mes amis, pour leur soutien indéfectible, leur écoute approfondie et leurs encouragements authentiques. Leur présence, qu'elle soit tangible ou spirituelle, a renforcé ma confiance en moi-même lors des moments de doute, rendant ainsi cette expérience plus humaine et gratifiante.

Finalement, je tiens à exprimer toute ma gratitude envers ma famille. Leur amour sans faille, leur patience et leur soutien moral constant ont été ma fondation durant toute cette traversée. Leur confiance, leur soutien continu et leurs encouragements tout au long du processus m'ont permis de trouver la force et le calme nécessaires pour réaliser ce projet avec succès. Ce triomphe leur revient tout autant qu'à moi, et j'aimerais leur dédier ce travail en signe de ma profonde gratitude.

Résumé

Ce document offre une analyse d'un système de reconnaissance des gestes fondé sur la technologie **Lin-LED**[1], un appareil novateur qui se sert de capteurs infrarouges pour interpréter les mouvements de la main avec exactitude. Le but est de détecter en direct les signaux provenant de photodiodes, qui captent la réflexion de la lumière infrarouge produite par les mouvements de l'utilisateur devant LinLED. C'est à l'aide d'un microcontrôleur Teensy 4.1 que les données ont été collectées pour cette étude. Les signaux analogiques émis par les photodiodes ont été convertis en numérique grâce à Arduino IDE, puis ils ont été prétraités, visualisés et stockés à l'aide de Matlab. Un jalon important a été l'élaboration d'une base de données expérimentale : **20 participants** de divers âges ont chacun accompli **4 gestes**, répétés au moins une dizaine de fois, ce qui a conduit à la collecte de **1034 échantillons**. Ces informations ont par la suite servi à l'entraînement et à l'évaluation des modèles d'apprentissage automatique. Comme les signaux bruts étaient bruités, plusieurs techniques de filtrage ont été appliquées afin de les nettoyer tout en conservant les informations utiles : filtre passe-bas de Butterworth, filtre de Savitzky-Golay et moyenne exponentielle mobile (EMA). Une fois les signaux préparés, des caractéristiques statistiques ont été extraites pour entraîner différents modèles d'apprentissage automatique. Les modèles testés (Random Forest, K-Nearest Neighbors, Perceptron Multi-Couche et HistGradientBoosting) ont été évalués à l'aide de matrices de confusion, de la précision et du temps d'inférence. Les résultats sont encourageants : le modèle HistGradientBoosting, associé au filtrage EMA, atteint une précision de **98 %**. Le modèle Random Forest obtient également d'excellentes performances, avec une précision de **96,62 %** et un temps de réponse très court (**0,019 ms** par geste). Ces mesures ont été réalisées sur une machine équipée de Windows 11 (64 bits), d'un processeur Intel Core i7-1355U (13^e génération, 1.70 GHz), d'une carte graphique intégrée Intel Iris Xe Graphics et de 16 Go de RAM. L'efficacité observée sur cette configuration standard confirme le potentiel d'une utilisation en temps réel, notamment dans des systèmes embarqués. Ces travaux ouvrent ainsi la voie à des applications concrètes de la reconnaissance gestuelle, par exemple dans le domaine automobile ou dans l'interaction homme-machine. Une application directe pourrait être le contrôle sans contact des tableaux de bord, via des interfaces naturelles, intuitives et réactives.

En résumé, ce projet démontre la faisabilité d'une solution combinant capteurs infrarouges, traitement du signal et apprentissage automatique, adaptée aux contraintes du temps réel et de l'embarqué.

Mots-clés : Reconnaissance gestuelle, capteurs infrarouges, acquisition et traitement du signal, filtrage, apprentissage automatique, classification, analyse de données, microcontrôleur, systèmes embarqués, temps réel.

1 Introduction

1.1 Contexte du projet

Avec l'avancement des technologies digitales, l'interaction homme-machine (IHM) s'est transformée en un défi crucial pour la conception de systèmes plus intuitifs et faciles d'accès. L'idée est de faciliter une interaction plus instinctive avec la machine, en recourant à des moyens déjà connus tels que la voix, le contact tactile ou les mouvements. La reconnaissance gestuelle, parmi ces méthodes, offre plusieurs bénéfices : absence de contact tactile, diminution de l'érosion du matériel, meilleure propreté (surtout dans les environnements médicaux ou publics) et facilité d'usage améliorée. L'industrie automobile est un bon exemple de ce potentiel : l'intégration de capteurs gestuels offre aux conducteurs la possibilité de gérer certaines fonctionnalités sans quitter des yeux la route, ce qui renforce ainsi la sécurité et enrichit l'expérience de l'utilisateur (voir Figure 2).

Le projet **LinLED** [1] s'inscrit dans ce cadre, proposant une méthode novatrice de reconnaissance gestuelle reposant sur des capteurs infrarouges. L'idée est claire : la main renvoie la lumière produite par des LED, et cette renvoyée est détectée par des photodiodes. Cette technologie présente de nombreux avantages : un délai de latence minime (approximativement 1 ms), une consommation énergétique réduite, une structure à la fois légère et résistante, et par-dessus tout, elle est insensible aux variations entre les utilisateurs et ne requiert pas de réentraînement du système.

Comparé à d'autres alternatives telles que celles proposées par **Neonone** (neonode.com), qui utilisent des lasers et des traitements plus complexes entraînant une latence accrue, LinLED opte pour un design minimaliste et performant, idéal pour les systèmes embarqués dotés de ressources restreintes.

Néanmoins, le défi principal demeure de conserver un délai d'interaction minime entre l'action et la réaction, condition essentielle pour assurer la fluidité et la fiabilité d'une interaction en temps réel.



FIGURE 2 – Exemple d'interface sans contact intégrée dans un tableau de bord.

1.2 Présentation de l'Institut des Sciences du Mouvement (ISM)

L'Institut des Sciences du Mouvement représente une unité de recherche hybride affiliée à l'Institut des Sciences Biologiques (INSB) du Centre National de la Recherche Scientifique CNRS, en collaboration avec l'Aix-Marseille Université (AMU). L'ISM rassemble des chercheurs, enseignants et ingénieurs en doctorat, qui sont répartis sur divers lieux. Trois importantes équipes de recherche structurent le laboratoire :

- **BioMécanique & Bio-Ingénierie (BMI)**,
- **Dynamiques Comportementales & Cognition (DynamiCC)**,
- **Systèmes Bio-Inspirés (SBI)**, au sein de laquelle s'est déroulé mon stage.

Mission et expertise

L'ISM, en intégrant des disciplines telles que la biomécanique et la robotique dans son étude du mouvement vivant, vise à établir un lien entre la recherche fondamentale et les applications pratiques, y compris le sport, la robotique et les transports. Pour réaliser ses tâches, l'ISM se fonde sur cinq plateformes technologiques avancées, parmi lesquelles :

- △ le centre de Réalité Virtuelle (CRVM),
- △ la plateforme TechnoSport,
- △ l'arène de vol méditerranéenne (AVM).

L'équipe Systèmes Bio-Inspirés (SBI)

J'ai effectué mon stage dans l'équipe (SBI), sous la direction de Jean-Marc Linares et Stéphane Viollet. La méthode de cette équipe s'appuie sur un principe essentiel : *reconstruire pour comprendre*. Elle élabore des solutions novatrices pour l'ingénierie, en observant et en s'inspirant de la nature. L'équipe structure ses recherches autour de deux axes principaux qui se complètent mutuellement :

- **Biorobotique** : conception de robots et de capteurs bio-inspirés capables de se déplacer sans GPS, de percevoir leur environnement ou de s'adapter à des conditions complexes.
- **Mécanismes bio-inspirés** : développement de structures mécaniques, d'actionneurs ou de modèles issus de la morphogenèse et de la biomécanique, en lien avec les technologies de l'industrie du futur.

1.3 Objectifs du projet

Ce projet de fin d'études s'inscrit dans le cadre du développement de **LinLED** [1], un dispositif de détection gestuelle sans contact conçu à l'ISM. Ce système repose sur une ligne de photodiodes associées à des LED infrarouges, permettant de détecter avec une grande précision différents types de gestes grâce à la réflexion de la lumière infrarouge sur la main placée devant LinLED. L'objectif principal de ce projet est de concevoir une chaîne complète de traitement permettant :

- ◊ l'acquisition fiable des signaux analogiques issus des photodiodes via un microcontrôleur (**Teensy 4.1**) et leur numérisation via **Arduino IDE**,
- ◊ le prétraitement et la visualisation des signaux sous **Matlab**,
- ◊ l'extraction de caractéristiques pertinentes pour la classification des gestes,
- ◊ la mise en œuvre et l'évaluation de modèles d'apprentissage automatique performants sous **Python**, capables de fonctionner en temps réel dans des conditions proches de l'usage embarqué.

L'enjeu est de garantir une détection fiable et rapide des gestes tout en maintenant une faible complexité algorithmique, compatible avec des systèmes embarqués à ressources limitées.

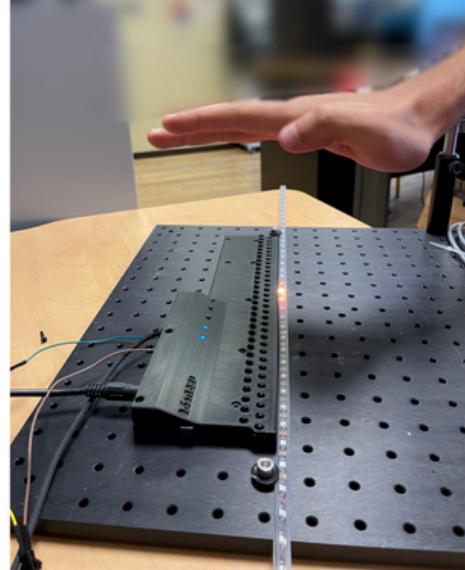


FIGURE 3 – Système LinLED accompagné d'un retour visuel par LED.

C'est dans ce cadre que se pose la problématique suivante :

1.4 Problématique

LinLED constitue une solution prometteuse pour l'interaction gestuelle dans des contextes critiques comme l'automobile, où la réactivité et la fiabilité sont essentielles.

Cependant, pour que cette technologie soit pleinement exploitable dans des environnements à ressources limitées, plusieurs verrous technologiques doivent être levés :

- ◊ Assurer une reconnaissance gestuelle précise et robuste en temps réel ;
- ◊ Respecter les contraintes strictes liées au calcul embarqué : faible consommation énergétique, usage limité de la mémoire et de la puissance de traitement ;
- ◊ Minimiser la latence de bout en bout entre le geste et la réponse du système, tout en conservant une complexité algorithmique faible.

Problématique centrale :

Comment concevoir et implémenter un système de reconnaissance gestuelle infrarouge à la fois rapide, fiable et peu gourmand en ressources, adapté aux contraintes du traitement embarqué en temps réel ?

Pour répondre à cette problématique, deux volets complémentaires structurent ce projet de fin d'études :

- **Volet théorique** : Étude et mise en œuvre d'un nouvel algorithme d'apprentissage supervisé à l'aide d'exemples, développé au sein de l'équipe *Systèmes Bio-Inspirés*, reposant sur des outils d'algèbre linéaire ;
- **Volet applicatif** : Déploiement de cet algorithme sur une tâche de reconnaissance de gestes, en collaboration avec le groupe **Stellantis**, via l'utilisation du capteur optique LinLED, dans des conditions proches de l'usage embarqué.

1.5 Organisation du rapport

Dans le but d'aborder cette question, ce rapport est organisé de la façon suivante :

La section 2 présente un état de l'art des approches existantes en matière d'apprentissage embarqué et de reconnaissance gestuelle.

La section 3 décrit les ressources matérielles et logicielles mobilisées au cours du projet.

La section 4 détaille la méthodologie adoptée ainsi que les étapes de développement des algorithmes.

La section 5 expose les résultats obtenus, en termes de performance et de consommation des ressources.

La section 6 analyse ces résultats, en mettant en lumière les limites identifiées et les pistes d'amélioration envisagées.

La section 7 met en avant ma contribution personnelle au projet, à travers les tâches réalisées et les solutions proposées.

Enfin, la section 8 conclut le rapport en résumant les apports du travail réalisé et propose des perspectives pour la suite du projet.

2 État de l'art

Au cours des dernières années, les interfaces basées sur les gestes ont radicalement modifié la façon dont les usagers interagissent avec les systèmes numériques. Avec l'avènement de capteurs divers, la détection de mouvements sans contact, c'est-à-dire sans manipulation d'appareil ou de surface, est désormais réalisable. Ces solutions offrent des possibilités prometteuses, surtout en matière d'accessibilité, de domotique ou d'interfaces homme-machine (IHM). Dans cette optique, le projet LinLED est mis en œuvre, avec l'objectif de développer une solution intégrée, indépendante, légère et fiable. Afin de bien comprendre les défis, il est crucial d'examiner les principales méthodologies en place, leurs rendements, ainsi que leurs restrictions, notamment dans un environnement restreint tel que celui d'un microcontrôleur.

On peut distinguer deux catégories majeures : les interfaces qui font appel à des capteurs physiques dédiés, et celles qui tirent parti des capteurs existants dans les systèmes.

Interfaces utilisant des capteurs physiques

Ces systèmes s'appuient sur des capteurs externes ou portés pour capter les mouvements. Ils sont souvent précis, mais peuvent être coûteux, intrusifs ou peu adaptés à une intégration embarquée.

Capteurs portés sur la peau

Kim et al. [2] ont développé un capteur nanomesh imprimé directement sur la peau. Il détecte de très légers mouvements cutanés, permettant une reconnaissance fine de gestes comme le pincement. Grâce au *meta-learning*, le système s'adapte à chaque utilisateur. Cependant, son caractère intrusif limite son usage quotidien.

Capteurs intégrés aux smartphones

AirWare, proposé par Lohia et al. [3], exploite les haut-parleurs, microphones et capteurs IR déjà présents dans les smartphones. En analysant l'effet Doppler, le système atteint plus de 80 % de précision pour des gestes simples. Toutefois, sa dépendance à la position du smartphone et la nécessité de calibrations réduisent sa robustesse.

Bracelets à LED infrarouges

Maereg et al. [4] ont conçu un bracelet contenant 36 capteurs IR mesurant la réflexion sur le poignet. Leur système atteint 98 % de précision en temps réel. Malgré ces performances, la complexité matérielle et la consommation énergétique sont des obstacles à une intégration embarquée.

Reconnaissance par image infrarouge

Mantecon et al. [5] utilisent directement les images infrarouges pour détecter les gestes, sans modélisation de squelette. Cette méthode est robuste aux occlusions, mais très gourmande en ressources, donc peu adaptée aux microcontrôleurs.

Caméra fisheye pour suivi 3D

DeepFisheye, développé par Park et al. [6], utilise une caméra ultra-grand-angle placée sous une surface. Le système suit les doigts en 3D avec une erreur moyenne de 20 mm. Sa précision est bonne, mais il nécessite du matériel spécialisé et de puissants algorithmes.

Capteurs IR basse résolution

À l'opposé, Yamato et al. [7] ont montré que des capteurs infrarouges très simples suffisent pour reconnaître des gestes basiques. Leur système, basé sur des modèles d'apprentissage légers, propose un bon compromis entre coût, consommation et efficacité ; une approche proche de celle de LinLED.

Approches sans capteurs spécifiques

Ces méthodes exploitent les capteurs déjà disponibles dans les appareils (caméras, lasers, etc.). Elles sont plus faciles à déployer, mais souvent plus exigeantes en ressources de calcul.

Caméras RGB ou à profondeur

De nombreuses études utilisent des flux vidéo pour la reconnaissance de gestes. La revue d'Al-Assam et al. [8] analyse plus de 100 travaux entre 2012 et 2022. Certaines approches atteignent 97 % de précision grâce aux réseaux de neurones convolutifs ou récurrents. Ces méthodes nécessitent cependant des ressources importantes (GPU, Edge TPU) et sont sensibles aux variations de lumière ou d'arrière-plan.

Systèmes hybrides multi-capteurs

Jiang et al. [9] identifient une tendance vers l'hybridation : combiner plusieurs types de capteurs (optiques, électromagnétiques, mécaniques) pour plus de robustesse. Ces systèmes sont puissants mais difficiles à miniaturiser et à adapter à l'embarqué.

Technologie laser sans contact

Des solutions comme Neonode utilisent des faisceaux laser pour détecter les gestes à distance. Ces produits sont rapides et efficaces, mais leur coût élevé et leur nature propriétaire en limitent l'intégration dans des systèmes personnalisés. Neonode est un concurrent direct de LinLED, avec des approches techniques très différentes.

Positionnement du projet LinLED

Face aux limites des solutions actuelles, **LinLED** propose une approche originale : utiliser des photodiodes et des LED infrarouges pour capter la réflexion lumineuse, sans imagerie ni contact. Cette solution offre plusieurs avantages :

- **Respect de la vie privée** : aucun enregistrement visuel.
- **Faible latence** : traitement local et rapide.
- **Simplicité matérielle** : peu de composants, facile à intégrer.
- **Consommation réduite** : adaptée aux systèmes embarqués.

Le principal défi reste l'adaptation aux variations individuelles (position, morphologie, vitesse du geste). C'est précisément dans ce contexte que ce projet cherche à apporter une solution, via un traitement optimisé du signal et des modèles de classification efficaces.

Le projet est présenté plus en détail sur son site officiel, et l'article scientifique associé est disponible à l'adresse suivante : https://amu.hal.science/hal-04245112/file/LinLed_low.pdf. Un premier prototype a également été dévoilé lors de la conférence ACM ICMI 2023, accompagné d'un poster en ligne.

3 Matériaux

Les éléments matériels et logiciels utilisés dans ce projet ont permis de mettre en place l'ensemble du système de reconnaissance gestuelle basé de LinLED, depuis la phase de collecte des données jusqu'à l'entraînement du modèle. Cette partie est organisée en deux sections principales :

- ◊ La maquette d'acquisition, dédiée à la capture des signaux gestuels à l'aide de LinLED.
- ◊ La configuration logicielle, utilisée pour l'implémentation de l'algorithme, la création de la base de données et l'entraînement du modèle de classification.

3.1 Maquette d'acquisition

3.1.1 Le dispositif LinLED

LinLED (Linear LED-based interface) [1] est une technologie innovante de détection de gestes sans contact, développée à l'ISM de Marseille. Contrairement aux systèmes utilisant des caméras, LinLED ne capture pas d'images ni ne transfère de données personnelles, ce qui respecte la vie privée et le RGPD (voir ici). Son traitement entièrement analogique permet une détection rapide et précise avec un très faible délai, idéale pour suivre les gestes en temps réel.

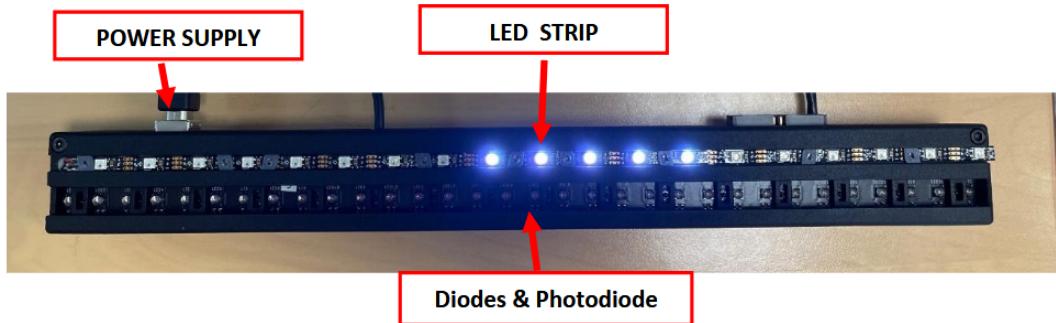


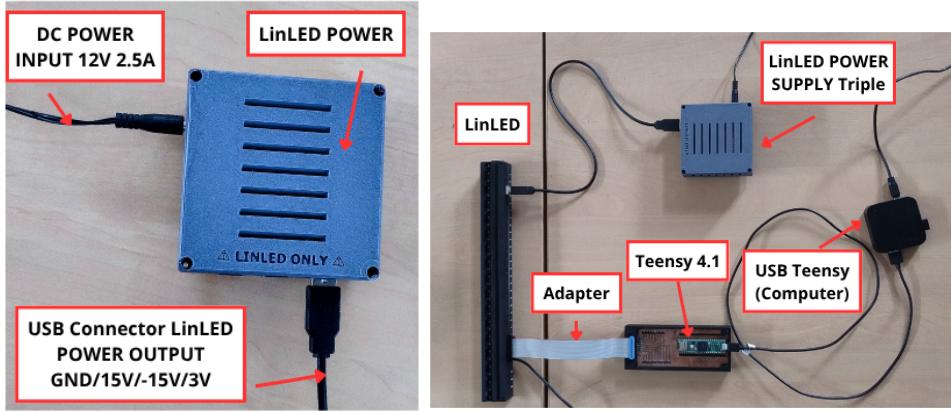
FIGURE 4 – Dispositif LinLED

- ◊ **Longueur** : 320 mm
- ◊ **Dimensions** : 320 × 35 × 20 mm
- ◊ **Portée de détection** : Jusqu'à 500 mm
- ◊ **LED (Infrarouge)** : 32
- ◊ **Photodiodes (capteurs)** : 16
- ◊ **Temps de réponse maximal** : 400 μ s
- ◊ **Résolution maximale** : 320 μ m
- ◊ **Technologie** : Infrarouge
- ◊ **Type d'interaction** : Doigt, main, objet réfléchissant les infrarouges



(a) Aperçu du prototype LinLED

(b) Face arrière du LinLED



(a) Bloc d'alimentation LinLED

(b) Câblage du prototype

FIGURE 6 – Illustrations du prototype LinLED

Important : Le prototype LinLED utilise un connecteur USB qui, bien qu'il ressemble à un port standard, est réservé exclusivement à son bloc d'alimentation spécifique 6a. Ce dernier fournit des tensions particulières indispensables au fonctionnement du système. Il est donc crucial de ne jamais connecter LinLED à un ordinateur, un chargeur USB classique ou tout autre appareil grand public via ce port. Une mauvaise connexion pourrait endommager irrémédiablement le prototype. En résumé, ce port USB doit uniquement être utilisé pour connecter le bloc d'alimentation dédié, comme illustré à la figure 6b.

3.1.2 Principe de fonctionnement de LinLED

Le système LinLED s'inspire du poisson électrique [10], un animal capable de percevoir son environnement grâce à un champ électrique. De manière analogue, LinLED exploite un principe sensoriel innovant reposant sur la réflexion de la lumière infrarouge.



FIGURE 7 – Poisson électrique

Le dispositif **LinLED** détecte les gestes sans contact à partir de l'émission et de la réception de lumière infrarouge. Il est composé de LED émettant en continu un rayonnement infrarouge et de capteurs (photodiodes) chargés de le recueillir. Lorsqu'un objet réfléchissant, tel qu'une main, traverse le champ d'émission, une partie du rayonnement est réfléchie et captée par les photodiodes. L'analyse de l'intensité du signal réfléchi permet de déterminer la position de l'objet avec une précision millimétrique, rendant possible la localisation et le suivi du mouvement d'un doigt ou d'une main au-dessus de LinLED.

Propriétés clés de la somme pondérée de gaussiennes

- **Chevauchement nécessaire :** Pour que le système fonctionne bien, les gaussiennes (voir ici) des photodiodes (figure 9) doivent se chevaucher d'au moins 15 %. Ce chevauchement permet à la fonction résultante W_s (somme pondérée) de devenir presque linéaire dans la zone centrale.
- **Monotonie et quasi-linéarité :** Si le chevauchement est suffisant, W_s devient monotone croissante et presque linéaire avec la position. Cela permet une estimation très précise, même si la distance entre les photodiodes est relativement grande.

- **Lien avec l'hyperacuité :** L'hyperacuité [11] désigne la capacité à détecter des déplacements plus fins que le pas spatial du capteur (ici, la distance entre photodiodes). Grâce à la quasi-linéarité de la somme pondérée, le dispositif peut localiser un doigt avec une précision supérieure à l'espacement physique des photodiodes.

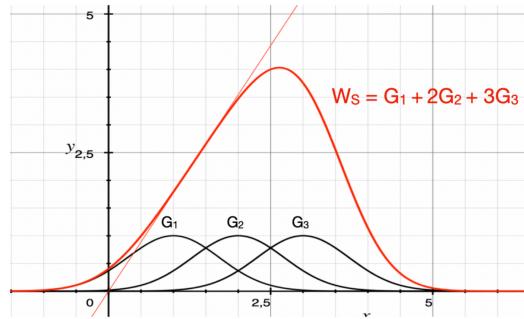


FIGURE 8 – Illustration des gaussiennes pondérées (Source)

En résumé : la somme pondérée de gaussiennes [1] chevauchantes transforme un réseau discret de capteurs en une fonction continue quasi-linéaire, expliquant la capacité d'hyperacuité du dispositif LinLED.

Structure et modulation

LinLED repose sur un réseau linéaire de 32 LEDs infrarouges, modulées à 30 kHz, alternant avec 16 photodiodes (figure 9). La modulation à 30 kHz permet de filtrer efficacement la lumière ambiante non modulée (lumière naturelle, lampes, autres LED), réduisant le bruit et les interférences.

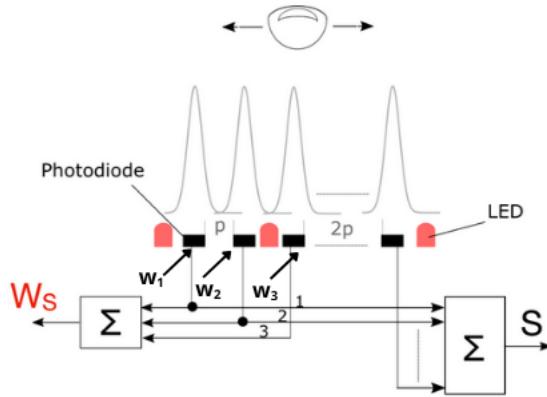


FIGURE 9 – Disposition des LEDs et photodiodes (Source)

Traitement analogique et filtrage

Chaque photodiode capte la lumière infrarouge réfléchie par un objet en mouvement (main, doigt) devant LinLED à une certaine distance variant de 10 à 40 cm environ. Le signal réfléchi, faible et souvent noyé dans le bruit, est filtré par un amplificateur à verrouillage de phase (*lock-in amplifier*) accordé à la fréquence de modulation. Ce filtrage sélectif rejette les perturbations lumineuses ambiantes et délivre un signal propre et précis.

Somme, pondération et estimation de position

Les 16 signaux analogiques w_1, w_2, \dots, w_{16} provenant des photodiodes sont d'abord traités par deux amplificateurs sommateurs.

La somme simple $S = \sum_{i=1}^{16} w_i$ mesure la lumière réfléchie globale, indiquant la présence ou non d'un objet proche.

La somme pondérée $W_s = \sum_{i=1}^{16} i \cdot w_i$ prend en compte la position physique de chaque photodiode pour estimer la position horizontale de l'objet.

À partir de ces quantités, on a :

$$X = \frac{W_s}{S} = \frac{\sum_{i=1}^{16} i \cdot w_i}{\sum_{i=1}^{16} w_i} \quad (\text{position horizontale estimée})$$

$$Y = \frac{S}{16} = \frac{1}{16} \sum_{i=1}^{16} w_i \quad (\text{intensité moyenne réfléchie})$$

Ici, X donne la position latérale de l'objet, tandis que Y renseigne sur sa proximité.

Avantages du traitement analogique

Ce traitement entièrement analogique présente plusieurs bénéfices :

- **Faible latence** : réponse quasi instantanée (<1 ms), idéale pour les interactions en temps réel.
- **Robustesse au bruit** : grâce à la modulation et au filtrage lock-in, le signal reste stable en environnement lumineux variable.
- **Efficacité énergétique et calculatoire** : limite la charge sur le traitement numérique, facilitant l'intégration dans des systèmes embarqués.

3.1.3 Acquisition des signaux et connexion avec la Teensy 4.1

Après avoir présenté le principe de fonctionnement du système **LinLED** et ses sorties analogiques, il est important de détailler leur exploitation via une plateforme matérielle adaptée.

Brochage de la sortie analogique

Le connecteur situé à l'arrière du LinLED (figures 10 et 5b) permet de récupérer les données brutes issues des photodiodes. Il s'agit d'un connecteur 20 broches, réparties en deux rangées de 10, avec un pas standard de 2,54 mm. Chaque photodiode est connectée à une sortie délivrant une tension analogique proportionnelle à l'intensité de la lumière infrarouge réfléchie. Ces tensions sont mesurées par le microcontrôleur **Teensy 4.1** (figure 11b) pour une analyse en temps réel.



FIGURE 10 – Brochage de la sortie analogique (Analog OUTPUT)

Moyenne analogique	GND	S2	S4	S6	S8	S10	S12	S14	S16
Somme analogique	GND	S1	S3	S5	S7	S9	S11	S13	S15

TABLE 1 – Affectation des broches du connecteur de sortie analogique

Les broches du connecteur ont les fonctions suivantes :

1. **Signaux analogiques (S1 à S16)** : transportent les tensions analogiques (comprises entre 0 et 3,3 V) produites par les 16 photodiodes, représentant l'intensité lumineuse détectée par chaque capteur.

2. **Moyenne** : fournit la moyenne des signaux, représentant l'intensité globale détectée par l'ensemble des photodiodes.
3. **Somme pondérée normalisée** : donne une estimation de la position latérale de l'objet, calculée comme un centre de gravité des signaux réfléchis.
4. **Masse (GND)** : référence commune essentielle pour la stabilité et la fiabilité des mesures.

Ce connecteur (figure 10) offre un accès complet aux mesures du LinLED, permettant une lecture détaillée via les sorties individuelles ou une acquisition simplifiée grâce aux signaux moyens et pondérés.

Exploitation des signaux analogiques

Les sorties analogiques du système **LinLED** sont directement connectées aux entrées analogiques (ADC) du microcontrôleur **Teensy 4.1**, assurant un traitement rapide et précis des données.

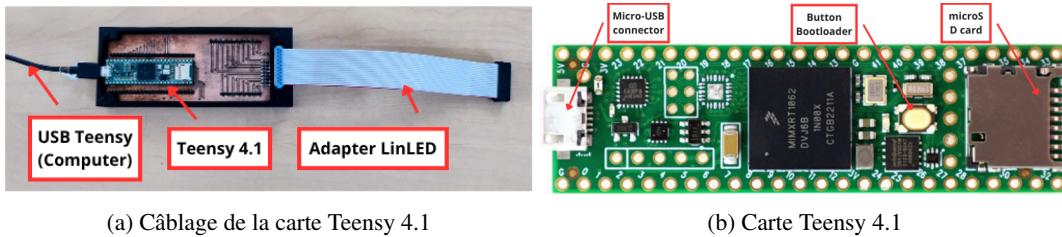


FIGURE 11 – Présentation de la carte Teensy 4.1

Les broches de sortie analogique du LinLED (figure 10) sont connectées aux entrées analogiques de la carte *Teensy 4.1* (figure 11b). La liaison des masses (GND) entre les deux cartes assure une référence commune indispensable à la stabilité et à la cohérence des mesures. La *Teensy 4.1*, sélectionnée pour sa compatibilité matérielle, dispose de nombreuses entrées analogiques, d'une résolution allant jusqu'à 12 bits, et d'une capacité d'acquisition rapide.

3.2 Configuration machine

Pour assurer cohérence, stabilité et reproductibilité, toutes les étapes (du développement logiciel à l'analyse des données et aux tests d'inférence) ont été réalisées sur une même machine. Ce choix limite les variations liées à des différences matérielles et garantit une homogénéité des résultats, indispensable pour valider un système de détection basé sur l'analyse de signaux analogiques.

3.2.1 Matériel utilisé

- LinLED composé :
 - 32 LED infrarouges modulées à 30 kHz : alignées linéairement, elles émettent un faisceau infrarouge dirigé vers la zone d'interaction.
 - 16 photodiodes intercalées entre les LED : elles captent la lumière réfléchie par les objets situés jusqu'à 40 cm au-dessus, avec une résolution allant jusqu'à 1 mm, bien inférieure à l'espacement physique des photodiodes (égal à 2 cm donc 20 fois meilleure).
 - Amplificateurs à verrouillage de phase : chaque photodiode est reliée à un circuit de démodulation analogique permettant de supprimer le bruit ambiant (éclairage artificiel, lumière naturelle) et d'extraire le signal utile avec une très faible latence (~1 ms).
 - Sorties analogiques : une par photodiode, plus deux supplémentaires dédiées à la somme pondérée et à la somme totale.
 - Carte microcontrôleur Teensy 4.1 : choisie pour sa haute fréquence d'horloge (jusqu'à 600 MHz), sa compatibilité avec l'environnement Arduino, et sa capacité à numériser rapidement les signaux analogiques avant de les transmettre au système hôte via USB.

- Connecteur 20 broches : permet de connecter de manière stable l'ensemble des signaux entre la carte LinLED et l'électronique d'acquisition.
- Règle graduée (ou centimètre) : utilisée lors d'expérimentation pour mesurer la distance.
- Carte SD : pour le stockage local des données d'acquisition.
- Alimentation stabilisée DC : pour garantir un courant stable aux LED et à l'électronique.
- Connecteurs, câbles USB et ports : pour faciliter les connexions.

3.2.2 Ordinateur utilisé

L'ensemble des étapes de l'acquisition des données jusqu'à l'apprentissage automatique, en passant par le traitement et la visualisation a été réalisé sur un seul et même ordinateur portable dont :

- **Système** : Windows 11 (64 bits)
- **Processeur** : Intel Core i7-1355U (13^e génération, 1.70 GHz)
- **Carte graphique** : Intel Iris Xe Graphics (intégrée)
- **RAM** : 16 Go

Cette configuration s'est révélée suffisante pour traiter les données en temps réel, exécuter les modèles de *machine learning*, et générer les visualisations nécessaires à l'évaluation des performances. L'absence de GPU dédié n'a pas été un frein, les algorithmes utilisés étant peu gourmands en ressources.

3.2.3 Environnement logiciel

Le développement s'est appuyé sur un ensemble cohérent d'outils logiciels :

- **Arduino IDE** : utilisé pour programmer la carte Teensy 4.1, avec l'extension Teensyduino dédiée au support matériel. Les tensions analogiques sont lues via l'ADC intégré, et la communication avec l'ordinateur se fait par liaison série (`Serial`). La bibliothèque `Wire` a servi pour les échanges (en I2C) avec certains modules périphériques.
- **MATLAB** : choisi pour sa rapidité à générer des visualisations claires et interactives. Il a été employé pour le suivi en temps réel, le prétraitement des données et leur sauvegarde.
- **Python** : utilisé pour l'analyse approfondie et les phases de machine learning. Les bibliothèques mobilisées incluent :
 - ◊ `numpy`, `pandas` : pour manipuler et structurer les données ;
 - ◊ `matplotlib`, `seaborn` : pour des visualisations avancées et la création de graphiques exploitables dans un rapport ;
 - ◊ `scikit-learn` : pour l'entraînement et l'évaluation de modèles prédictifs.

Cette combinaison logicielle a permis d'assurer la fiabilité des résultats tout en limitant les différences liées aux configurations matérielles. Elle constitue une base solide pour les étapes suivantes d'inférence et d'évaluation du système.

4 Méthodes

Cette section détaille l'ensemble de la procédure mise en œuvre pour l'acquisition et le traitement des données, depuis la constitution de la base jusqu'à l'entraînement des modèles. Les résultats issus de cette méthodologie seront présentés séparément dans la section 5.

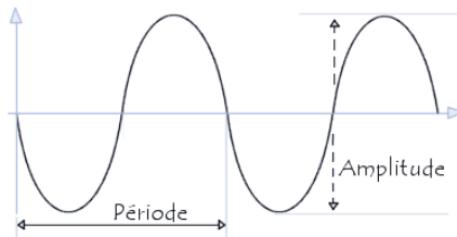
4.1 Création et mise en forme de la base de données

Une base de données a été constituée dans des conditions d'acquisition proches de celles attendues en phase d'inférence, afin d'assurer une bonne généralisation des performances.

4.1.0.1 Définition du signal

Un **signal** est une information représentée par une variation mesurable dans le temps. Dans notre cas, il s'agit d'une tension électrique générée par les photodiodes, évoluant en fonction de la lumière IR réfléchie. Pour mieux caractériser un signal, plusieurs propriétés fondamentales doivent être prises en compte :

- **Amplitude** : valeur maximale atteinte par le signal, exprimée en volts (V), elle reflète l'intensité des variations.
- **Fréquence (f)** : nombre de cycles ou d'oscillations par seconde, exprimé en hertz (Hz).
- **Période (T)** : durée d'un cycle complet du signal, exprimée en secondes (s). Elle est l'inverse de la fréquence : $T = \frac{1}{f}$



On distingue principalement deux types de signaux :

◊ **Signal analogique** : signal continu dont la valeur peut varier de manière fluide entre deux bornes (exemple. : 0 à 3,3 V). C'est la forme brute issue des photodiodes de LinLED. Il reflète fidèlement l'intensité lumineuse reçue, mais il est souvent sujet à des perturbations (bruit, interférences, etc.). (Analog-Signal)

◊ **Signal numérique** : résultat de la conversion du signal analogique à l'aide d'un convertisseur analogique-numérique. Le signal est alors discrétisé, sous forme d'une suite de valeurs codées (exemple : 0 à 1023 pour un ADC 10 bits). Cette forme facilite le traitement, le stockage et l'analyse sur ordinateur. (Digital-Signal)

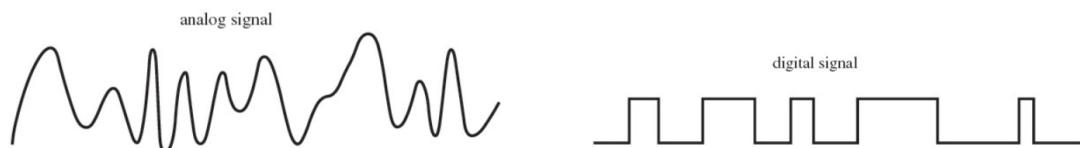


FIGURE 12 – Exemple de signal analogique et numérique, illustrant la conversion du signal continu en valeurs discrètes. Source

La conversion analogique-numérique (ADC) est une étape clé : elle transforme un signal continu en une suite d'échantillons discrets. Cette opération est réalisée à une certaine **fréquence d'échantillonnage** (ici 200 Hz), ce qui signifie que le signal est mesuré toutes les 5 ms. Un choix judicieux de cette fréquence est crucial pour respecter le théorème de Nyquist-Shannon [12] et éviter la perte d'information.

4.1.1 Base de données

Dans ce projet, une base de données (disponible ici) a été constituée à partir de gestes capturés dans des conditions d'acquisition proches de celles prévues lors de l'inférence.

4.1.1.1 Acquisition des signaux avec Arduino

Le dispositif LinLED génère des signaux analogiques en réponse aux variations de lumière infrarouge, réfléchie par les mouvements des mains. Ces signaux sont captés par des photodiodes, puis transmis à un microcontrôleur Teensy 4.1, qui effectue une lecture temps réel des valeurs analogiques et les transmet via un port série à un ordinateur pour traitement et visualisation.

L'algorithme présenté ci-dessous (également disponible ici) illustre ce processus d'acquisition :

Algorithm 1 Lecture et transmission de données analogiques avec Teensy 4.1

```
1: Constantes : nChannels = 18, nSamples = 10000, SAMPRATE = 200 Hz    ▷ Définit le nombre
   de canaux, la taille du tampon et la fréquence d'échantillonnage
2: Variables : data[nChannels][nSamples], writeIndex = 0           ▷ Tableau pour stocker les
   données et index circulaire d'écriture
3: Initialisation :
   ◊ Démarrer la communication série à 250000 bauds    ▷ Permet d'envoyer les données au PC via
     USB
   ◊ Attendre que le port série soit prêt
   ◊ Configurer l'ADC :
     • Résolution de 10 bits (valeurs entre 0 et 1023)
     • Pas d'averaging matériel (1 seule mesure brute)
     • Vitesse de conversion et d'échantillonnage très rapide
   ◊ Définir les 18 broches analogiques utilisées (A0 à A17)
4: while le programme est en cours d'exécution do                      ▷ Boucle principale de l'acquisition
5:   Étape 1 : Lire les 18 entrées analogiques
6:   for ch = 0 à nChannels - 1 do
7:     Lire la valeur analogique sur la broche adc_pins[ch]
8:     Stocker la valeur dans data[ch][writeIndex]    ▷ On enregistre la valeur dans le tampon
   circulaire
9:   end for
10:  Étape 2 : Envoyer une ligne de données en CSV
11:  for ch = 0 à nChannels - 1 do
12:    Envoyer data[ch][writeIndex] via le port série
13:    if ch < nChannels - 1 then
14:      Ajouter une virgule
15:    end if
16:  end for
17:  Terminer la ligne avec un saut de ligne (newline)    ▷ On obtient une ligne CSV lisible par
   MATLAB
18:  Étape 3 : Incrémenter writeIndex
19:  Si writeIndex atteint nSamples, le remettre à 0          ▷ Gestion circulaire du tampon
20:  Étape 4 : Attendre 5 ms    ▷ Permet de respecter la fréquence d'échantillonnage de 200 Hz
21: end while
```

Remarque : Le format de sortie CSV permet une compatibilité directe avec MATLAB, facilitant l'analyse et la visualisation des signaux en temps réel.

4.1.1.2 Visualisation et enregistrement avec MATLAB

MATLAB reçoit les données en continu via le port série. Les données sont systématiquement enregistrées sous forme de fichiers texte (.txt), servant de base pour l'analyse et l'exploitation ultérieure.

Le fonctionnement détaillé est présenté dans l'algorithme suivant (lien vers le code source) :

Algorithm 2 Acquisition et labellisation en temps réel de signaux analogiques (MATLAB)

```
1: Initialisation des variables globales :
    ◊ current_label ← "Neutral"
    ◊ enregistrement_actif ← true      ▷ L'enregistrement commence comme actif par défaut
    ◊ Préparer les fichiers de sortie pour l'enregistrement

2: Demander l'ID du participant          ▷ Permet de créer un dossier spécifique pour chaque session utilisateur

3: Créer / ouvrir le fichier .txt

4: if fichier vide then
5:     Écrire l'en-tête ("A1,...,A18,label")
6: end if

7: Définir une fonction de rappel clavier :      ▷ Permet de changer de label ou de démarrer/arrêter l'enregistrement avec le clavier
    ◊ Selon la touche pressée, mettre à jour current_label
    ◊ Si touche [A] : basculer entre pause et reprise
    ◊ Fermer le fichier en cours (si ouvert)
    ◊ Ouvrir (ou créer) un fichier correspondant au nouveau label
    ◊ Si ce fichier est vide, y ajouter l'en-tête CSV

8: Configurer le port série          ▷ Connexion à la carte Teensy sur COM6 à 250000 bauds

9: Créer une fenêtre graphique pour l'affichage en direct
    ◊ Afficher en temps réel les valeurs du canal 17 (Sn)          ▷ Pour un retour visuel immédiat
    ◊ Lier les touches clavier à la fonction de rappel

10: while la figure reste ouverte do

11:     if des données sont disponibles sur le port série then
12:         Lire une ligne (chaîne de 18 valeurs)
13:         Convertir la ligne en tableau numérique      ▷ Chaque ligne représente un instant avec 18 mesures
14:         if le tableau contient bien 18 valeurs then
15:             Ajouter la valeur A17 au graphique          ▷ Affichage en direct pour l'utilisateur
16:             Déterminer le label à enregistrer
17:             if A17 ∈ [5, 30] et A18 ∈ [0, 25] then
18:                 Label forcé = "Neutral"      ▷ Filtrage automatique en fonction de l'état des signaux
19:             else
20:                 Utiliser le label choisi par l'utilisateur (current_label)
21:             end if
22:             if enregistrement actif then
23:                 Écrire la ligne (valeurs + label) dans le fichier du label
24:             end if
25:             Écrire la ligne (valeurs + label) dans le fichier global      ▷ Sauvegarde complète même si non enregistrée par label
26:         end if
27:     end if
28: end while
29: Fermser les fichiers ouverts          ▷ Nettoyage à la fin de l'exécution
```

Remarque : Ce script MATLAB permet de visualiser et d'étiqueter les données en temps réel, en liant chaque ligne de données à un label contrôlé par l'utilisateur. Le canal 17 est affiché en direct pour aider à la décision de labellisation.

Exemple de données enregistrées

Les signaux capturés par LinLED sont enregistrés ligne par ligne dans des fichiers *TXT*, chaque ligne correspondant à un instant précis et contenant les valeurs des 18 canaux suivies du label associé.

Pour illustrer, le tableau ci-dessous montre un extrait des données acquises lors de différents mouvements de la main. Les colonnes A1 à A18 représentent les mesures des capteurs, et la dernière colonne correspond au label assigné :

A1	A2	A3	A4	A5	A17	A18	label
6	10	19	2	0	17	12	Neutral
7	12	21	4	0	19	13	Neutral
8	12	20	4	0	17	12	Neutral
9	13	6	0	32	106	85	Click
10	13	7	0	36	92	77	Click
9	12	8	0	38	89	73	Click
10	14	10	1	40	84	67	Click
10	14	11	3	40	76	61	Click
10	13	13	7	41	70	57	Click
8	13	28	15	16	29	24	Neutral
7	15	29	14	14	29	22	Neutral
8	16	28	11	10	26	20	Neutral
8	14	27	12	9	26	20	Neutral

Exemple de signal obtenu

Ci-dessous, un exemple de signaux capturés lors d'un mouvement lent de la main de gauche à droite devant LinLED :

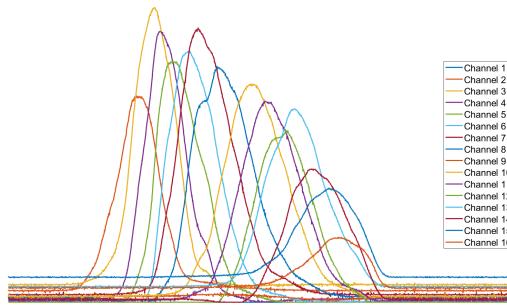
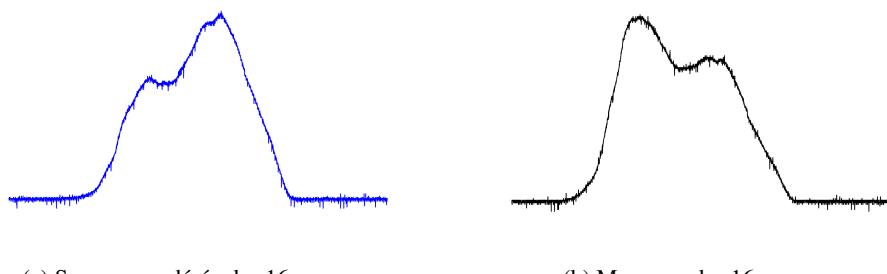


FIGURE 13 – Signal issu des 16 canaux (capteurs) pendant un mouvement de la main



(a) Somme pondérée des 16 canaux

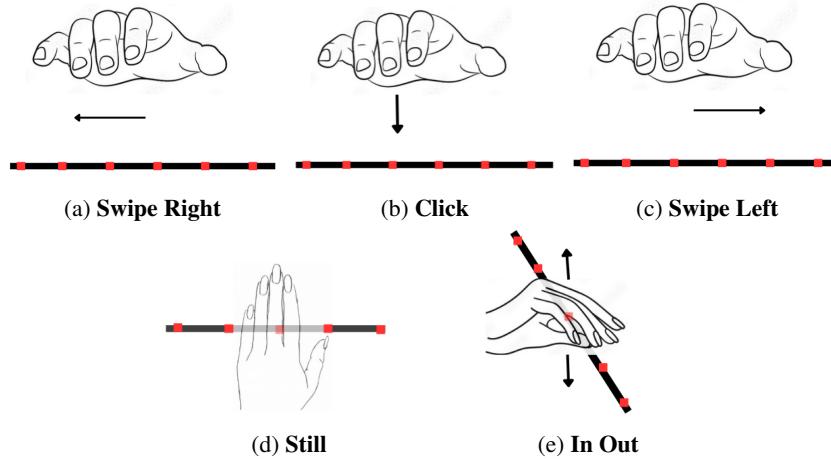
(b) Moyenne des 16 canaux

Remarque : Au début du projet, c'est l'Arduino qui s'occupait de l'enregistrement des données, mais à cause de limitations dans la gestion de sa mémoire, des données étaient parfois perdues ou corrompues. Pour garantir la qualité et la fiabilité des signaux acquis, nous avons confié cette tâche à MATLAB qui gère mieux le flux série en temps réel et assure un stockage fiable des données.

4.1.1.3 Nature des gestes enregistrés

La base comprend plusieurs types de gestes représentatifs des interactions attendues :

- **Click** : mouvement rapide de la main vers le dispositif, simulant une sélection.
- **Swipe Right** : déplacement horizontal de la main de droite à gauche, utilisé pour la navigation.
- **Swipe Left** : déplacement horizontal dans le sens inverse.
- **Still** : main immobile devant le capteur pendant 1 à 5 secondes, représentant une pause.
- **In Out** : mouvement d'approche puis de retrait de la main devant le dispositif.



4.1.1.4 Qualité et diversité des données

Les données ont été enregistrées lors de plusieurs sessions dans un environnement semi-contrôlé, sans imposer de contraintes trop strictes, à part demander aux participants d'exécuter des gestes précis. Chaque personne était assise confortablement, et le dispositif LinLED était placé sur une table, à environ 80 cm du sol. Les enregistrements se sont déroulés à l'intérieur, avec une lumière naturelle. L'étude a porté sur 20 participants de l'Institut des Sciences du Mouvement (ISM), dont 75 % avaient moins de 30 ans. Chaque geste a été répété 10 fois par participant. Voici la répartition des gestes enregistrés :

- | | |
|--|---|
| <ul style="list-style-type: none"> ◊ Click : 206 séquences ◊ InOut : 210 séquences ◊ Still : 203 séquences | <ul style="list-style-type: none"> ◊ SwipeLeft : 207 séquences ◊ SwipeRight : 208 séquences |
|--|---|

Remarque : Théoriquement, avec 20 participants réalisant 5 gestes répétés 10 fois chacun, on s'attendait à un total de 200 séquences par geste. Toutefois, de légères variations ont été observées, certains participants ayant effectué plus de répétitions que prévu initialement.

4.1.1.5 Règle heuristique complémentaire

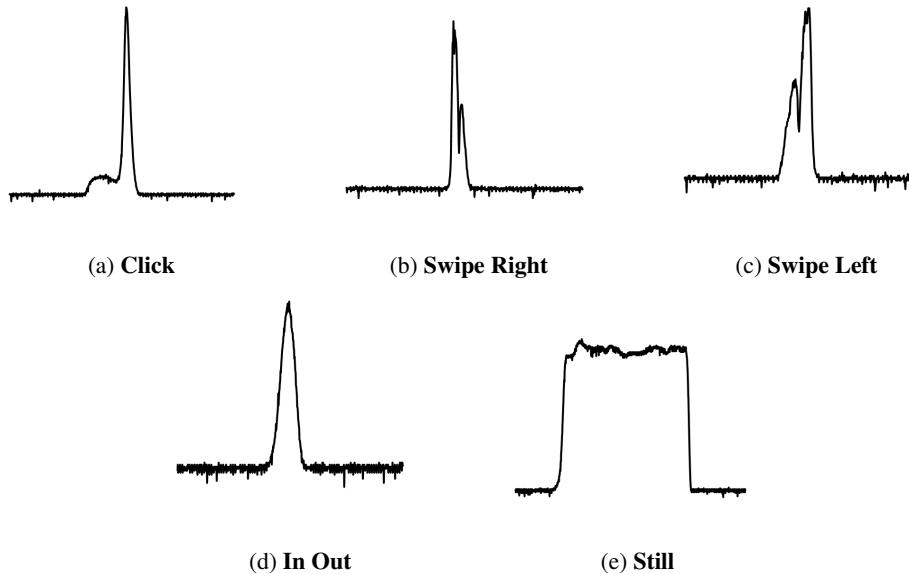
Pour mieux détecter les moments sans geste, une règle simple a été ajoutée dans le script MATLAB ; pour définir cette règle, plusieurs tests ont été réalisés :

- ◊ un enregistrement de 10 minutes sans aucun mouvement devant LinLED, pour observer le bruit naturel du signal,
- ◊ un test de 45 minutes en conditions variées, toujours sans interaction volontaire,
- ◊ une session d'environ une heure avec quelques petits mouvements involontaires, pour simuler un état proche du calme.

Ces essais ont permis de trouver des seuils caractéristiques. Quand les signaux restent dans ces plages, MATLAB applique automatiquement l'étiquette **Neutral**, indiquant qu'aucun geste n'est détecté. Cela permet d'éviter les erreurs d'étiquetage et d'améliorer la qualité globale des données.

Signaux bruts des différents gestes

La somme pondérée des signaux bruts obtenus pour chaque geste est présentée ci-dessous :



4.2 Apprentissage automatique

Nous avons entraîné de modèles d'apprentissage automatique pour que le système puisse apprendre à reconnaître les gestes automatiquement à partir des signaux enregistrés.

4.2.0.1 Qu'est-ce que l'apprentissage automatique ?

L'apprentissage automatique (ou *machine learning*) [13] est un domaine de l'intelligence artificielle qui consiste à apprendre à une machine à accomplir une tâche à partir d'exemples, plutôt que de la programmer explicitement pour chaque situation.

On peut l'imaginer comme un élève : au lieu de recevoir une consigne précise pour chaque cas, la machine observe des exemples, en déduit des régularités, puis utilise ce qu'elle a appris pour traiter de nouvelles situations.

Ce processus se déroule généralement en deux grandes étapes :

- **Apprentissage** : C'est la première étape. On fournit au modèle un grand nombre d'exemples déjà connus. Par exemple, si l'on veut qu'il reconnaît des gestes, on lui montre différentes données correspondant à des gestes, en précisant à chaque fois de quel geste il s'agit. Le modèle analyse ces exemples, cherche ce qu'ils ont en commun et apprend à faire la différence entre eux. Un peu comme un élève qui apprend à reconnaître un geste en observant sa forme ou son mouvement. Une fois cette étape terminée, le modèle est dit *entraîné* : il a appris à associer des données à des catégories.
- **Prédiction** : Après l'apprentissage, le modèle peut maintenant recevoir de nouvelles données qu'il n'a jamais vues. Grâce à ce qu'il a appris, il va essayer de deviner à quelle catégorie ces données appartiennent. Plus les exemples utilisés pendant l'apprentissage étaient nombreux, variés et bien choisis, plus les prédictions du modèle seront fiables. Sinon, il faudra revoir l'étape d'apprentissage pour qu'il puisse s'améliorer.

En résumé : le *machine learning* consiste à apprendre à un modèle à reconnaître des éléments à partir d'exemples. Dans un premier temps, on lui montre de nombreux cas connus (**apprentissage**), puis il utilise ce qu'il a retenu pour analyser de nouvelles situations et proposer des réponses (**prédiction**). Ce fonctionnement est illustré dans la figure suivante :

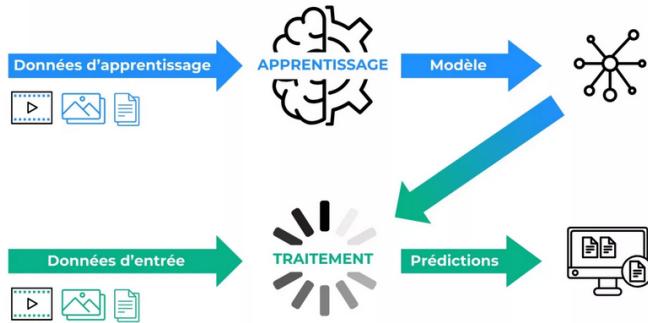


FIGURE 17 – Schéma du processus d'apprentissage automatique (Source)

Métriques d'évaluation

Pour évaluer la performance du modèle, plusieurs indicateurs complémentaires sont utilisés. Chacun donne un aperçu différent de la qualité des prédictions.

- **Accuracy (ou exactitude)** : c'est la part des prédictions correctes parmi toutes les prédictions. Ça donne une idée globale de la performance, mais peut être trompeuse si une classe est beaucoup plus fréquente que les autres. Elle se calcule comme ça :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- ◊ TP : vrais positifs, le modèle a bien reconnu un geste présent,
- ◊ TN : vrais négatifs, le modèle a correctement identifié l'absence de geste,
- ◊ FP : faux positifs, le modèle a prédit un geste alors qu'il n'y en avait pas,
- ◊ FN : faux négatifs, le modèle a raté un geste réellement effectué.

- **Précision** : indique, parmi toutes les fois où le modèle a prédit une certaine classe, combien étaient correctes. Elle est définie par :

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

- **Rappel (ou sensibilité)** : mesure la capacité du modèle à détecter les gestes réellement présents. Plus le rappel est élevé, moins le modèle rate de gestes :

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

- **F1-score** : combine précision et rappel en une seule valeur, particulièrement utile lorsque les classes sont déséquilibrées, car il équilibre les erreurs de type faux positifs et faux négatifs. Il est défini par :

$$F_1 = \frac{2 \times \text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

4.2.0.2 Les différents types d'apprentissage

Selon la quantité et la nature des données disponibles, plusieurs méthodes d'apprentissage automatique peuvent être utilisées :

- **Apprentissage supervisé** : chaque donnée d'entrée a une étiquette qui indique sa catégorie. Le modèle apprend à associer les données à la bonne catégorie.
- **Apprentissage non supervisé** : il n'y a pas d'étiquettes pour les données. Le modèle cherche à regrouper les données qui se ressemblent naturellement.
- **Apprentissage semi-supervisé** : une partie seulement des données a des étiquettes. Le modèle utilise ces exemples pour apprendre, tout en profitant aussi des données non étiquetées.
- **Apprentissage auto-supervisé** : le modèle génère lui-même des étiquettes à partir des données, par exemple en apprenant à deviner une partie du signal grâce à une autre.

4.2.0.3 Pourquoi l'apprentissage supervisé ?

Dans notre projet, chaque séquence de signaux est associée à un geste précis, ce qui justifie l'usage de l'*apprentissage supervisé* : le modèle s'entraîne à partir de données annotées pour apprendre à reconnaître les motifs propres à chaque geste et pouvoir ensuite classer correctement de nouvelles séquences, et leurs performances optimisées grâce à une recherche d'**hyperparamètres** (voir ici), afin d'ajuster la complexité, la robustesse et la vitesse d'apprentissage du modèle.

Random Forest

Le *Random Forest* [14] est un algorithme d'ensemble fondé sur l'utilisation de multiples arbres de décision [15]. Chaque arbre est entraîné sur un sous-échantillon aléatoire du jeu de données (technique appelée *bagging*). Lors de la prédiction, chaque arbre donne son avis (ou *vote*) pour une classe, et la classe obtenant le plus de votes est choisie comme prédiction finale, les principaux hyperparamètres testés incluent :

- ◊ `n_estimators` : nombre d'arbres dans la forêt. Plus ce nombre est élevé, plus le modèle est robuste, mais cela augmente aussi le temps de calcul.
- ◊ `max_depth` : profondeur maximale des arbres. Une faible profondeur évite le sur-apprentissage (*overfitting*), tandis qu'une grande profondeur permet de mieux modéliser des données.
- ◊ `min_samples_split` : nombre minimal d'échantillons requis pour diviser un nœud. Cela contrôle la croissance des arbres et agit comme un régulateur de complexité.
- ◊ `min_samples_leaf` : nombre minimal d'échantillons qu'une feuille doit contenir. Il empêche la création de feuilles trop spécifiques à un petit nombre de données, ce qui améliore la généralisation.

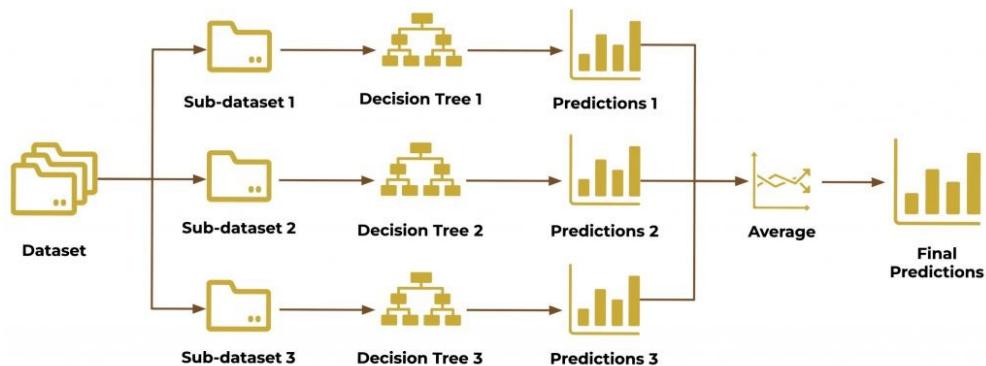


FIGURE 18 – Principe de fonctionnement d'un classifieur Random Forest (Source)

K-Nearest Neighbors (KNN)

L'algorithme *K-Nearest Neighbors* (KNN) [16] repose sur une intuition simple : une donnée est souvent similaire à ses voisins proches. Pour classer un nouvel exemple, KNN cherche les k points les plus proches dans les données d'entraînement (selon une distance, souvent euclidienne), puis lui attribue la classe la plus représentée parmi eux, les hyperparamètres [17] importants sont :

- ◊ k : nombre de voisins à considérer. Un petit k rend le modèle plus sensible au bruit, tandis qu'un grand k le rend plus stable mais moins réactif aux détails locaux.
- ◊ **weights** : méthode de pondération des voisins. `uniform` donne le même poids à tous les voisins, `distance` accorde plus d'importance aux voisins les plus proches.
- ◊ p : paramètre de la distance de Minkowski. $p = 1$ correspond à la distance de Manhattan (ou L1), tandis que $p = 2$ donne la distance euclidienne (ou L2). Ce choix influence la manière dont la proximité est mesurée.

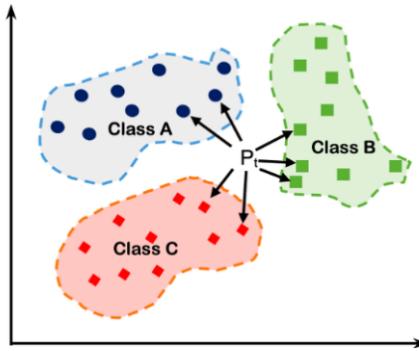


FIGURE 19 – Illustration du principe de KNN (Source)

HistGradientBoosting

Le *HistGradientBoosting* [18] est une version optimisée de l'algorithme de *Gradient Boosting*, proposée par la bibliothèque Scikit-learn. Cet algorithme construit plusieurs arbres de décision [15] de manière successive, chaque nouvel arbre visant à corriger les erreurs des arbres précédents, ses principaux hyperparamètres [17] sont :

- ◊ **learning_rate** : taux d'apprentissage. Il contrôle la contribution de chaque nouvel arbre. Une valeur faible augmente la stabilité mais nécessite plus d'arbres.
- ◊ **l2_regularization** : coefficient de régularisation L2. Il pénalise les arbres trop complexes pour limiter le surapprentissage. Une valeur de 0 désactive cette régularisation, tandis qu'une valeur plus élevée (par exemple 1) augmente la pénalité, rendant le modèle plus simple et plus général.
- ◊ **max_iter** : nombre total d'arbres construits (ou d'itérations). Plus il y a d'arbres, plus le modèle peut apprendre finement, mais au risque de surajuster si mal régulé.
- ◊ **max_depth** : profondeur maximale des arbres. Comme dans la Random Forest, elle limite la complexité de chaque arbre pour éviter l'overfitting.

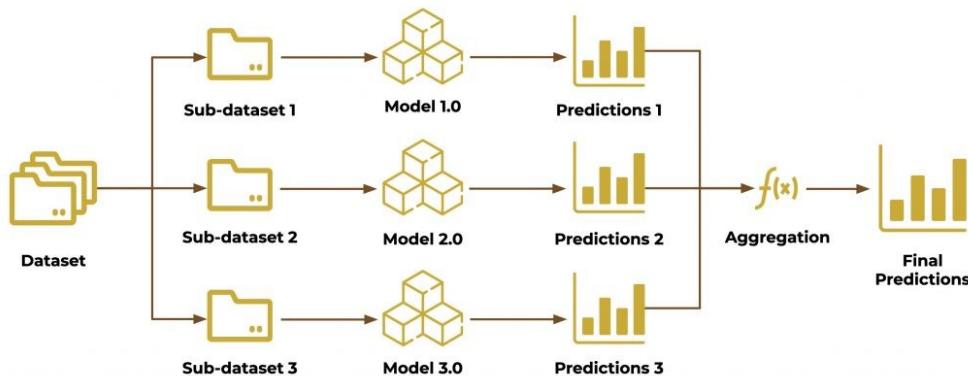


FIGURE 20 – Fonctionnement d'un modèle de type Boosting (Source)

MLP (Perceptron Multi-Couche)

Le Perceptron Multi-Couche [19], est un des réseaux de neurones [20] les plus classiques. Il est constitué de plusieurs couches de neurones entièrement connectées entre elles. Autrement dit, chaque neurone d'une couche envoie sa sortie à tous ceux de la couche suivante.

Un MLP est dit *multi-couche* (source) car il contient :

- ◊ une **couche d'entrée** : qui reçoit les caractéristiques du geste (ex. valeurs issues des photodiodes) ;
- ◊ une ou plusieurs **couches cachées** : où le réseau apprend à repérer des relations entre les données ;
- ◊ une **couche de sortie** : qui indique le geste détecté

Chaque neurone effectue une opération simple : il combine les valeurs qu'il reçoit, ajoute un biais, puis applique une fonction d'activation pour produire une sortie. En combinant des milliers de ces opérations, le réseau apprend à classer un geste à partir d'un ensemble de données brutes.

Où les hyperparamètres [17] agissent-ils dans le réseau ?

- **Nombre de couches cachées** : plus il y a de couches, plus le réseau peut apprendre des relations complexes entre les gestes. Mais trop de couches augmente le risque de surapprentissage.
- **Nombre de neurones par couche** : plus il y a de neurones, plus la couche peut représenter des gestes variés et complexes.
- **Fonction d'activation** (ex. ReLU) : elle introduit de la non-linéarité, indispensable pour que le réseau apprenne autre chose que des relations trop simples.
- **Taux d'apprentissage** : règle la vitesse à laquelle les poids du réseau s'ajustent. Trop élevé = instable, trop faible = lent.
- **Nombre d'époques** : correspond au nombre de fois où chaque donnée sert à entraîner le réseau. Trop peu = apprentissage insuffisant ; trop = risque de mémorisation.
- **Taille du batch** : indique combien d'exemples sont utilisés avant chaque mise à jour. Petits batches = rapides mais bruités ; grands batches = stables mais lourds à calculer.
- **Optimiseur** (ex. Adam) : c'est la méthode utilisée pour ajuster les poids de manière efficace en fonction des erreurs.
- **Régularisation** (ex. dropout) : désactive aléatoirement certains neurones pendant l'entraînement pour éviter que le réseau ne s'adapte trop aux données d'entraînement.

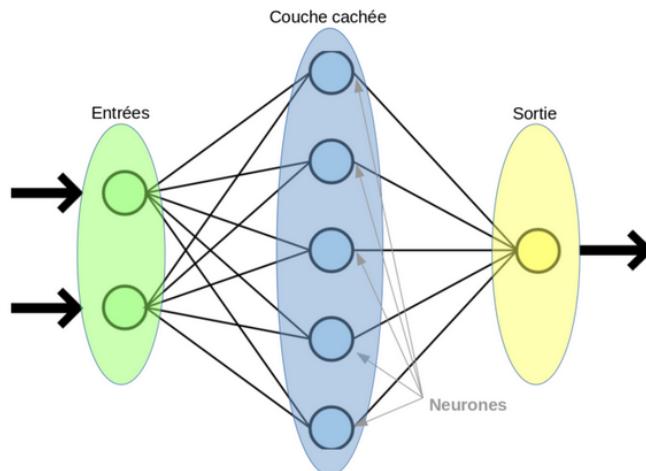


FIGURE 21 – Représentation simplifiée d'un réseau de neurones. (Source)

Algorithm 3 Pipeline de traitement des signaux (Python)

```
1: Chargement des fichiers .txt    ▷ Lecture des données brutes sauvegardées après enregistrement
2: for all fichiers dans le dossier do
3:   if le fichier contient les colonnes A1 à A18 et un label then
4:     Ajouter aux données valides           ▷ On garde seulement les fichiers bien formatés
5:   else
6:     Ignorer et enregistrer l'erreur ▷ Permet de diagnostiquer d'éventuels problèmes de collecte
7:   end if
8: end for
9: Filtrage des signaux
10: for all colonnes (A1 à A16) do
11:   Appliquer le filtre
12: end for
13: Segmentation des séquences      ▷ On découpe les signaux en blocs continus avec un même label
14: for all lignes du fichier do
15:   if le label reste constant then
16:     Ajouter la ligne à la séquence courante
17:   else
18:     if la séquence est suffisamment longue then
19:       Sauvegarder la séquence
20:     end if
21:     Commencer une nouvelle séquence avec le nouveau label
22:   end if
23: end for
24: Supprimer les séquences ayant le label "Neutral"
25: Extraction des caractéristiques
26: for all séquences valides do
27:   for all canaux A1 à A16 do
28:     Calculer : moyenne, minimum, maximum, écart-type
29:   end for
30:   Calculer la moyenne de A17 (intensité totale)
31:   Calculer la moyenne pondérée de A18 (position horizontale estimée)
32: end for
33: Préparation du jeu de données
34: Séparer X (caractéristiques) et y (labels)      ▷ X est l'entrée du modèle, y est la sortie attendue
35: Diviser en données d'entraînement et de test
36: Entraînement du modèle
37: Définir les hyperparamètres à tester
38: Lancer une recherche par grille avec validation croisée
39: Conserver le modèle ayant les meilleures performances moyennes
40: Évaluation du modèle
41: Prédire les classes sur les données de test
42: Calcule de précision et du temps moyen de prédiction
43: Matrice de confusion
```

4.3 Filtres

Qu'est-ce qu'un filtre ?

Un **filtre** est un dispositif ou un algorithme qui permet de sélectionner certaines fréquences d'un signal tout en atténuant les autres. Il sert principalement à améliorer la qualité des signaux en réduisant le bruit ou les composantes indésirables. Dans le domaine de l'électronique et du traitement du signal :

- **Audio** : séparer les graves, médiums et aigus pour les envoyer à différents haut-parleurs.
- **Acquisition de données** : éviter le repliement de spectre avant conversion analogique-numérique.

4.3.1 Filtre Passe-Bas

Le **filtre passe-bas** [21, 22] élimine les hautes fréquences d'un signal, souvent associées au bruit. En conservant uniquement les basses fréquences, il lisse le signal tout en préservant sa structure globale.

Mathématiquement, un filtre passe-bas simple peut être modélisé par une équation différentielle discrète, par exemple un filtre de Butterworth d'ordre n :

$$H(\omega) = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}}$$

où ω_c est la fréquence de coupure. voir ici

Ou, sous forme récursive (filtre passe-bas simple) :

$$y[t] = \alpha \cdot x[t] + (1 - \alpha) \cdot y[t - 1]$$

avec $0 < \alpha < 1$, $x[t]$ le signal d'entrée et $y[t]$ le signal filtré.

Avantages :

- ◊ Élimine les variations rapides (souvent du bruit) du signal.
- ◊ Fournit une sortie plus stable, facilitant l'analyse et la détection de tendances.

Rôle des paramètres :

- ◊ **Ordre** : détermine la raideur de la transition du filtre entre les fréquences conservées et atténuées. Un ordre plus élevé donne un filtrage plus net.
- ◊ **Fréquence de coupure (ω_c)** : fréquence au-delà de laquelle les composantes du signal sont fortement atténuées. Plus elle est basse, plus le filtrage est agressif.

Paramètres utilisés pour la figure 22b :

- ◊ Ordre du filtre : 3 ; fréquence de coupure : 0,01 Hz

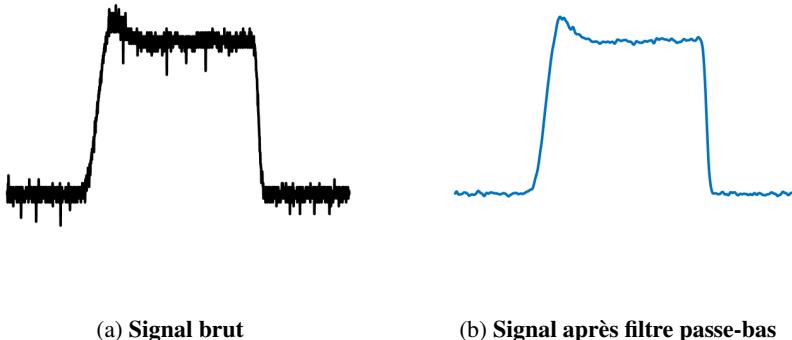


FIGURE 22 – Effet du filtre passe-bas sur la somme pondérée du signal du geste *Still*

4.3.2 Filtre de Savitzky-Golay

Le **filtre de Savitzky-Golay** applique une régression polynomiale locale sur une fenêtre glissante. Il est efficace pour lisser les données tout en conservant les pics, creux et la forme générale du signal.

Mathématiquement, la valeur filtrée au point t est obtenue par [23] :

$$y[t] = \sum_{k=-m}^m c_k \cdot x[t+k]$$

où $2m + 1$ est la taille de la fenêtre, et c_k sont les coefficients calculés pour minimiser l'erreur quadratique de la régression polynomiale d'ordre d sur cette fenêtre.

Avantages :

- ◊ Conserve les détails du signal (pics, creux) tout en réduisant le bruit.
- ◊ Préserve la forme générale sans décaler les données (pas de retard de phase).

Choix et rôle des paramètres :

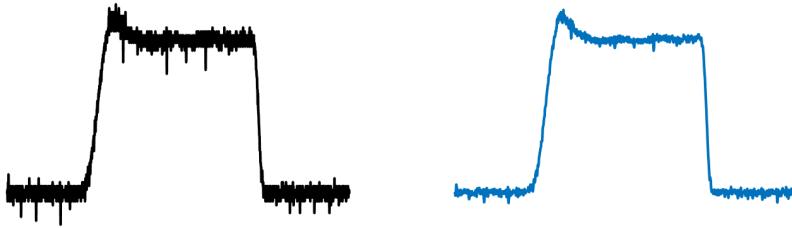
- ◊ ordre (ou degré) : détermine la complexité du polynôme utilisé pour approximer localement le signal. Un ordre plus élevé peut mieux suivre les variations rapides, mais risque aussi d'amplifier le bruit. À l'inverse, un ordre trop faible lisse trop et peut perdre des détails utiles.
- ◊ fenêtre : taille de l'intervalle glissant ($2m + 1$, obligatoirement impaire). Une fenêtre plus large produit un lissage plus fort, mais réduit la réactivité du filtre aux variations rapides.

À noter : l'ordre du polynôme doit toujours être strictement inférieur à la taille de la fenêtre ($d < 2m + 1$). En cas contraire, l'approximation devient instable voire impossible [24].

Paramètres utilisés pour la figure 23b :

- ◊ ordre : 3 ; fenêtre : 7

La figure suivante montre la somme pondérée du signal du geste *still* :



(a) Signal brut

(b) Après filtre Savitzky-Golay

FIGURE 23 – Effet du filtre Savitzky-Golay sur la somme pondérée du signal du geste *Still*

4.3.3 Filtre EMA (Exponential Moving Average)

Le **filtre EMA** (moyenne mobile exponentielle) est un filtre récursif qui attribue un poids décroissant aux anciennes valeurs du signal, comme décrit dans [25] et expliqué également sur Investopedia.

Il est défini par la relation récursive suivante (voir par exemple source) :

$$y[t] = \alpha \cdot x[t] + (1 - \alpha) \cdot y[t - 1] \quad ; \quad \alpha = \frac{2}{N + 1} \quad \begin{array}{l} \text{avec } N \text{ la taille de la fenêtre, } x[t] \text{ le signal d'entrée} \\ \text{et } y[t] \text{ le signal filtré.} \end{array}$$

Avantages :

- ◊ Il suit rapidement les changements récents du signal, ce qui est utile pour détecter les gestes en temps réel.
- ◊ Sa mise en œuvre est rapide, avec peu de calculs et peu de mémoire, ce qui le rend adapté aux systèmes embarqués.
- ◊ Il réduit le bruit sans déformer brutalement les variations du signal.

Rôle du paramètre :

- ◊ Fenêtre (N) : détermine la réactivité du filtre ; plus N est grand, plus le lissage est fort.

Paramètres utilisés pour la figure 24b :

- ◊ Fenêtre $N = 9$

La figure suivante montre la somme pondérée du signal du geste *still* :

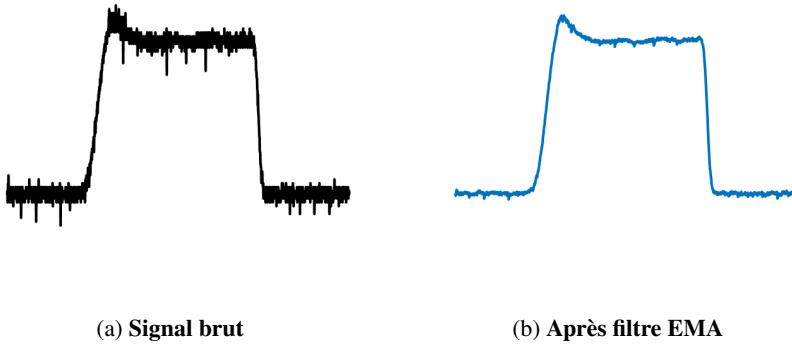


FIGURE 24 – Effet du filtre EMA sur la somme pondérée du signal du geste *Still*

Remarque : L'EMA (moyenne mobile exponentielle) et le filtre passe-bas sont **mathématiquement identiques** si l'on se limite à la forme récursive. Tous deux suivent la même équation :

$$y[t] = \alpha x[t] + (1 - \alpha)y[t - 1]$$

Cependant la différence ne réside pas dans l'équation, mais dans l'interprétation de α et le contexte :

Filtre passe-bas

- ◊ α est choisi pour fixer la fréquence de coupure ω_c .
- ◊ On analyse la réponse en fréquence : combien de hautes fréquences sont atténuées.
- ◊ Objectif : lisser ou filtrer un signal bruité selon sa fréquence.

EMA

- ◊ $\alpha = \frac{2}{N+1}$, avec N la taille de la fenêtre.
- ◊ Les valeurs récentes ont plus de poids, rendant le filtre réactif aux changements récents.
- ◊ Objectif : obtenir un indicateur lissé mais réactif aux variations.

La même équation est utilisée, mais la signification de α change selon le contexte : fréquence de coupure (ω_c) ou taille de fenêtre (N).

4.4 Segmentation des gestes

Pour analyser les mouvements captés par les capteurs, il est essentiel de transformer les données continues en **séquences distinctes**, chacune correspondant à un geste spécifique, et l'objectif est : de regrouper les lignes de données ayant le même label pour créer des séquences homogènes, cohérentes et filtrées des bruits ou gestes incomplets.

4.4.1 Principe de fonctionnement

La segmentation s'effectue sur un *DataFrame* [26] où chaque ligne correspond à un instant précis d'un geste, contenant :

- ◊ Une colonne `label` indiquant le type de geste.
- ◊ Les autres colonnes représentant les mesures des capteurs.

La segmentation se fait de manière séquentielle : la fonction parcourt chaque ligne du dataset et compare le label actuel avec celui de la séquence en cours. Si le label est identique, la ligne est ajoutée à la séquence existante. Dans le cas contraire, la séquence précédente est terminée et sauvegardée si elle contient un nombre de lignes suffisant. Une nouvelle séquence est alors initiée pour le nouveau label.

Cela permet de filtrer les gestes très courts ou incomplets, qui pourraient correspondre à du bruit ou à des erreurs de capture, améliorant ainsi la qualité des données.

4.4.2 Sortie de la fonction

La fonction retourne une **liste de tuples** :

- ◊ Le label de la séquence.
- ◊ Un *DataFrame* contenant toutes les lignes de cette séquence.

Exemple :

A1	A2	A3	A4	A18	label
0.1	0.2	1.5	1.0	1.2	Click
2.0	2.2	3.0	2.7	0.2	Click
0.1	0.2	0.5	1.0	2.2	Click
0.0	0.0	0.0	0.0	0.0	Neutral
0.0	0.0	0.0	0.0	0.0	Neutral
1.0	1.2	1.5	1.3	0.1	Click
0.1	0.2	2.5	1.5	1.7	Click

La fonction générera trois séquences distinctes :

1. Une séquence de type *Click* avec trois lignes.
2. Une séquence de type *Neutral* avec deux lignes.
3. Une séquence de type *Click* avec deux lignes.

4.4.3 Intérêt de l'extraction

La segmentation des gestes présente plusieurs bénéfices :

- ◊ Organisation efficace des données pour les algorithmes de classification.
- ◊ Isolation des mouvements cohérents, facilitant l'analyse détaillée.
- ◊ Amélioration de la qualité des données en éliminant les séquences trop courtes ou incohérentes.

Cette étape est donc **cruciale** pour garantir que les modèles d'intelligence artificielle reçoivent des séquences fiables et représentatives des gestes réels.

4.5 Extraction des caractéristiques des gestes

Après segmentation, nous extrayons des **caractéristiques statistiques**, permettant de résumer l'information de chaque séquence et de créer des descripteurs exploitables par les modèles d'apprentissage automatique.

4.5.1 Principe de la méthode

La fonction prend en entrée un *DataFrame* représentant la séquence, où les colonnes A1 à A18 sont les mesures des capteurs et chaque ligne une observation temporelle. Pour chaque colonne (A1 à A16), correspondant aux capteurs standard, la fonction calcule les statistiques suivantes :

- **Moyenne (mean)** : valeur moyenne du signal sur la séquence, donnant une idée du niveau moyen de l'activité.
- **Écart-type (std)** : mesure de la variation des valeurs, utile pour évaluer la régularité du geste.
- **Minimum (min)** : plus petite valeur observée dans la séquence.
- **Maximum (max)** : plus grande valeur observée dans la séquence.

Ces statistiques fournissent une représentation compacte mais informative du comportement du geste sur la durée. Les deux dernières colonnes, A17 et A18, représentent des signaux particuliers nécessitant un traitement spécifique :

- **A17_mean** : moyenne du signal, similaire aux colonnes précédentes.
- **A18_weighted** : moyenne pondérée du signal, pouvant représenter un indicateur combiné ou spécifique à l'analyse du geste.

4.5.2 Sortie de la fonction

La fonction retourne un **dictionnaire** où chaque clé correspond à une caractéristique extraite et chaque valeur à la mesure calculée. Par exemple :

- ◊ A1_mean : moyenne du capteur A1,
- ◊ A2_std : écart-type du capteur A2,
- ◊ A18_weighted : valeur moyenne pondérée de A18,
- ◊ etc.

4.5.3 Intérêt de l'extraction

- ◊ Réduction du volume de données tout en conservant l'information essentielle.
- ◊ Représentation de chaque geste sous forme de vecteur numérique exploitable par les modèles.
- ◊ Standardisation et amélioration de la précision des modèles.

Ainsi, cette étape constitue un pont essentiel entre les données et l'entraînement de modèles capables de reconnaître et classifier les gestes avec efficacité.

Commentaire :

Comme expliqué dans la section 4.1.1.1, nous collectons 10 000 échantillons par canal, soit 18 signaux analogiques acquis à une fréquence de 200 Hz équivalant à une mesure toutes les 5 ms. Chaque signal est numérisé à l'aide d'un convertisseur analogique-numérique (ADC) 10 bits, produisant des valeurs comprises entre 0 et 2500. Les données sont ensuite enregistrées sous forme de fichiers texte, avec 18 colonnes correspondant aux canaux (A1 à A18) et une colonne supplémentaire label indiquant le geste réalisé.

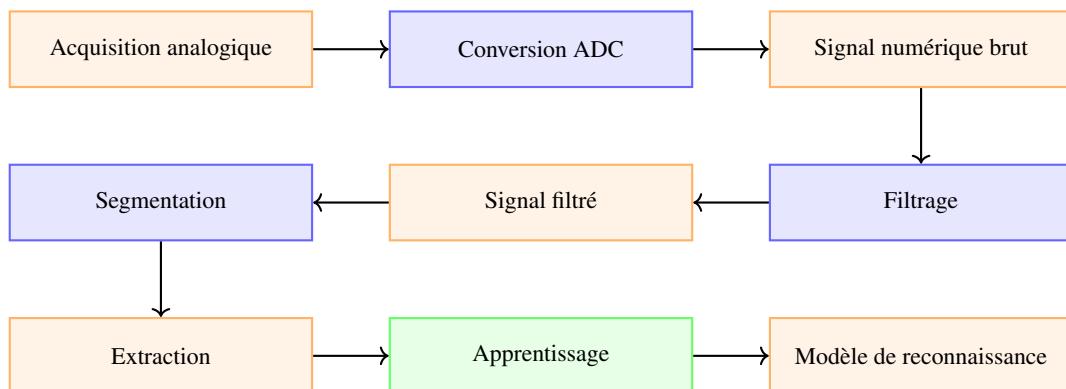


FIGURE 25 – Chaîne de traitement : de l'acquisition à l'apprentissage

5 Résultats

Dans cette section, nous présentons les résultats obtenus ici. Les métriques utilisées 4.2.0.1 :

- ◊ **Rapport de classification** : fournit trois mesures principales pour chaque classe :
 - **Précision.**
 - **Rappel (recall).**
 - **F1-score.**
- ◊ **Accuracy** : proportion de bonnes prédictions sur l'ensemble des données.
- ◊ **Matrice de confusion** : tableau qui compare les vraies classes et les classes prédites.

5.1 Méthode : Random Forest

Paramètres : seed = 42

Hyperparamètres :

- ◊ n_estimators = 50
- ◊ max_depth = None
- ◊ min_samples_split = 2
- ◊ min_samples_leaf = 1

5.1.1 Sans filtrage

5.1.1.1 Résultats numériques

Hyperparamètres	Precision (%)	Tps inférence par geste. (ms)
Non	88.41	0.137
Oui	88.41	0.082

TABLE 2 – Performances de Random Forest sans filtrage

5.1.1.2 Résultats visuels (sans filtrage ni hyperparamètres)

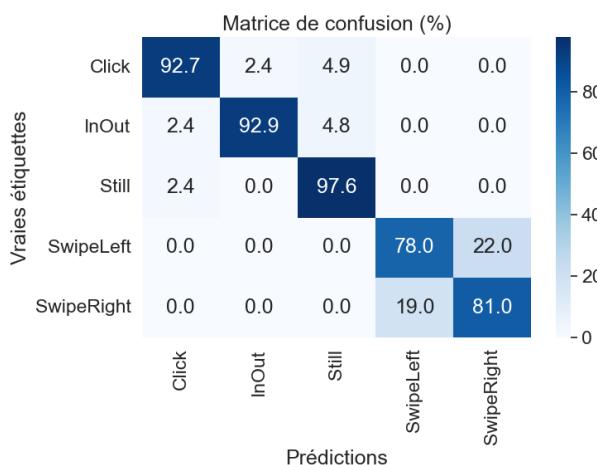


FIGURE 26 – Matrice de confusion de Random Forest sans filtrage

Classe	Precision	Rappel	F1-score	Support
Click	0.95	0.93	0.94	41
InOut	0.97	0.93	0.95	42
Still	0.91	0.98	0.94	41
SwipeLeft	0.80	0.78	0.79	41
SwipeRight	0.79	0.81	0.80	42
Accuracy			0.88	207
Macro avg	0.88	0.88	0.88	207
Weighted avg	0.88	0.88	0.88	207

TABLE 3 – Rapport de classification de Random Forest sans filtrage

Classe réelle	Classe prédictive	Nombre d'erreurs	Pourcentage d'erreurs (%)
Click	InOut	1	2.44
Click	Still	2	4.88
InOut	Click	1	2.38
InOut	Still	2	4.76
Still	Click	1	2.44
SwipeLeft	SwipeRight	9	21.95
SwipeRight	SwipeLeft	8	19.05

TABLE 4 – Tableau des erreurs de classification de Random Forest sans filtrage

5.1.2 Avec filtrage

Types de filtrage appliqués :

- **Filtre passe-bas** : ordre 3, fréquence de coupure $f_c = 0,9$ Hz
- **Savitzky-Golay** : ordre 3, fenêtre $N = 7$
- **EMA (Exponential Moving Average)** : fenêtre $N = 5$

5.1.2.1 Résultats numériques

Filtrage	Hyperpar-	Precision (%)	Tps d'infér- (ms/geste)
Pass-bas	Non	90.34	0.105
Pass-bas	Oui	89.37	0.031
Savitzky-Golay	Non	90.82	0.100
Savitzky-Golay	Oui	91.79	0.048
EMA	Non	96.62	0.092
EMA	Oui	96.14	0.075

TABLE 5 – Performances de Random Forest avec filtrage

5.1.2.2 Résultats visuels (filtrage EMA sans hyperparamètres)

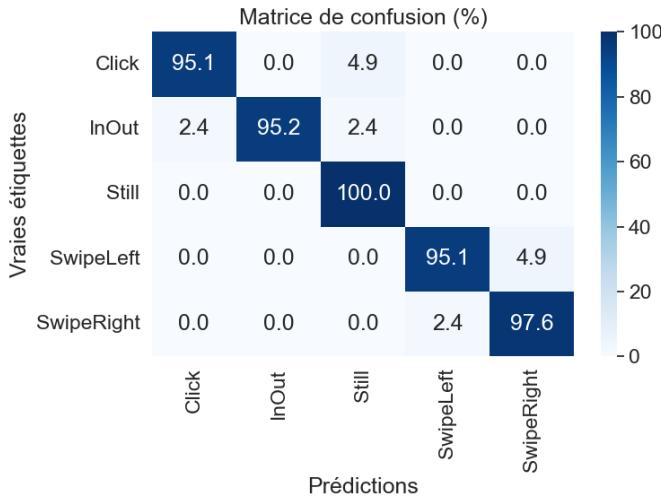


FIGURE 27 – Matrice de confusion de Random Forest sans filtrage

Classe	Précision	Rappel	F1-score	Support
Click	0.97	0.95	0.96	41
InOut	1.00	0.95	0.98	42
Still	0.93	1.00	0.96	41
SwipeLeft	0.97	0.95	0.96	41
SwipeRight	0.95	0.98	0.96	42
Accuracy			0.97	207
Macro avg	0.97	0.97	0.97	207
Weighted avg	0.97	0.97	0.97	207

TABLE 6 – Rapport de classification de Random Forest avec filtrage

Classe réelle	Classe prédite	Nombre d'erreurs	Pourcentage d'erreurs (%)
Click	Still	2	4.88
InOut	Click	1	2.38
InOut	Still	1	2.38
SwipeLeft	SwipeRight	2	4.88
SwipeRight	SwipeLeft	1	2.38

TABLE 7 – Tableau des erreurs de classification de Random Forest avec filtrage

5.2 Méthode : KNN

Hyperparamètres :

- ◊ weights fixée à "distance".
- ◊ p = 1.

5.2.1 Sans filtrage

5.2.1.1 Résultats numériques

n_neighbors	Hyperparamètres	Precision (%)	Tps infér- (ms/geste)
5	Non	80.68	0.103
5	Oui	84.06	0.167

TABLE 8 – Performances de KNN sans filtrage

5.2.1.2 Résultats visuels (sans filtrage avec hyperparamètres)

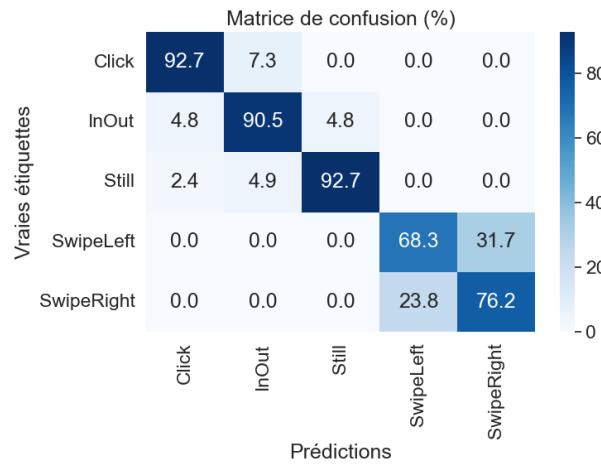


FIGURE 28 – Matrice de confusion de KNN sans filtrage

Classe	Précision	Rappel	F1-score	Support
Click	0.93	0.93	0.93	41
InOut	0.88	0.90	0.89	42
Still	0.95	0.93	0.94	41
SwipeLeft	0.74	0.68	0.71	41
SwipeRight	0.71	0.76	0.74	42
Accuracy			0.84	207
Macro avg	0.84	0.84	0.84	207
Weighted avg	0.84	0.84	0.84	207

TABLE 9 – Rapport de classification de KNN sans filtrage

Classe réelle	Classe prédictée	Nombre d'erreurs	Pourcentage d'erreurs (%)
Click	InOut	3	7.32
InOut	Click	2	4.76
InOut	Still	2	4.76
still	Clcik	1	2.44
Still	InOut	2	4.88
SwipeLeft	SwipeRight	13	31.71
SwipeRight	SwipeLeft	10	23.81

TABLE 10 – Tableau des erreurs de classification de KNN sans filtrage

5.2.2 Avec filtrage

Types de filtrage appliqués :

- **Filtre passe-bas** : ordre 3, fréquence de coupure $f_c = 0,9$ Hz
- **Savitzky-Golay** : ordre 3, fenêtre $N = 7$
- **EMA (Exponential Moving Average)** : fenêtre $N = 6$

5.2.2.1 Résultats numériques

Filtrage	n_neighbors	Hyperpar-	Précision (%)	Tps d'inf- (ms/geste)
Pass-bas	5	Non	79.71	0.197
Pass-bas	5	Oui	84.06	0.222
Savitzky-Golay	5	Non	81.64	0.088
Savitzky-Golay	5	Oui	83.09	0.111
EMA	5	Non	82.61	0.289
EMA	5	Oui	86.47	0.306

TABLE 11 – Performances de KNN avec filtrage

5.2.2.2 Résultats visuels (filtrage EMA avec hyperparamètres)

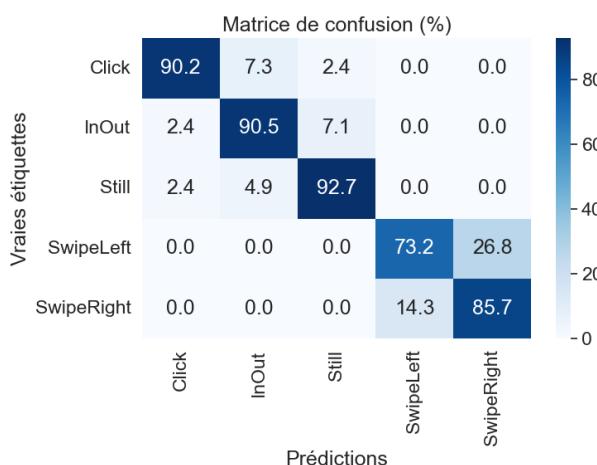


FIGURE 29 – Matrice de confusion de KNN avec filtrage

Classe	Précision	Rappel	F1-score	Support
Click	0.95	0.90	0.92	41
InOut	0.88	0.90	0.89	42
Still	0.90	0.93	0.92	41
SwipeLeft	0.83	0.71	0.76	41
SwipeRight	0.75	0.86	0.80	42
Accuracy			0.86	207
Macro avg	0.87	0.86	0.86	207
Weighted avg	0.87	0.86	0.86	207

TABLE 12 – Rapport de classification de KNN avec filtrage

Classe réelle	Classe prédictive	Nombre d'erreurs	Pourcentage d'erreurs (%)
Click	InOut	3	7.32
Click	Still	1	2.44
InOut	Click	1	2.38
InOut	Still	3	7.14
Still	Click	1	2.44
Still	InOut	2	4.88
SwipeLeft	SwipeRight	11	26.83
SwipeRight	SwipeLeft	6	14.29

TABLE 13 – Tableau des erreurs de classification de KNN avec filtrage

5.3 Méthode : HistGradientBoosting

Hyperparamètres :

- ◊ learning_rate = 0.5.
- ◊ max_iter= 50.
- ◊ max_depth fixée à None.
- ◊ l2_regularization = 0.

5.3.1 Sans filtrage

5.3.1.1 Résultats numériques

Hyperparamètres	Precision (%)	Tps inférence par geste. (ms)
Non	91.30	0.195
Oui	92.75	0.112

TABLE 14 – Performances de l’HistGradientBoosting sans filtrage

5.3.1.2 Résultats visuels (sans filtrage avec hyperparamètres)

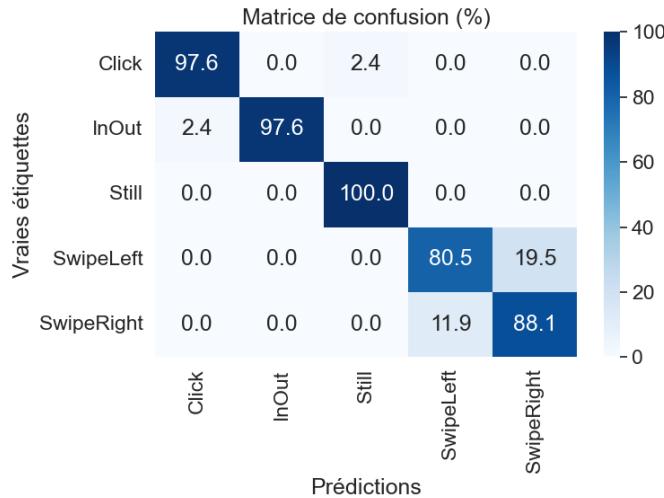


FIGURE 30 – Matrice de confusion de l’HistGradientBoosting sans filtrage

Classe	Précision	Rappel	F1-score	Support
Click	0.98	0.98	0.98	41
InOut	1.00	0.98	0.99	42
Still	0.98	1.00	0.99	41
SwipeLeft	0.87	0.80	0.84	41
SwipeRight	0.82	0.88	0.85	42
Accuracy			0.93	207
Macro avg	0.93	0.93	0.93	207
Weighted avg	0.93	0.93	0.93	207

TABLE 15 – Rapport de classification de l’HistGradientBoosting sans filtrage

Classe réelle	Classe prédictée	Nombre d’erreurs	Pourcentage d’erreurs (%)
Click	Still	1	2.44
InOut	Click	1	2.38
SwipeLeft	SwipeRight	8	19.51
SwipeRight	SwipeLeft	5	11.90

TABLE 16 – Tableau des erreurs de classification de l’HistGradientBoosting sans filtrage

5.3.2 Avec filtrage

Types de filtrage appliqués :

- **Filtre passe-bas** : ordre 3, fréquence de coupure $f_c = 0,9$ Hz
- **Savitzky-Golay** : ordre 5, fenêtre $N = 7$
- **EMA (Exponential Moving Average)** : fenêtre $N = 5$

5.3.2.1 Résultats numériques

Filtrage	Hyperparamètres	Précision (%)	Tps d'inférence (ms/geste)
Pass-bas	Non	92.75	0.193
Pass-bas	Oui	94.20	0.087
Savitzky-Golay	Non	93.24	0.181
Savitzky-Golay	Oui	92.75	0.069
EMA	Non	97.58	0.144
EMA	Oui	98.07	0.068

TABLE 17 – Performances avec filtre et réglage d'hyperparamètres

5.3.2.2 Résultats visuels (filtrage EMA avec hyperparamètres)

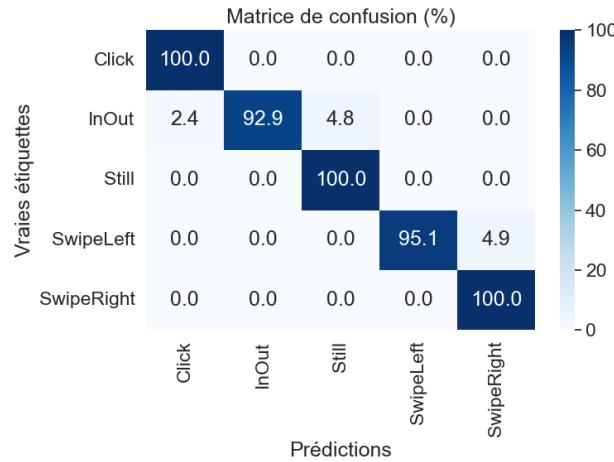


FIGURE 31 – Matrice de confusion de l'HistGradientBoosting avec filtre

Classe	Précision	Rappel	F1-score	Support
Click	0.98	1.00	0.99	41
InOut	1.00	0.93	0.96	42
Still	0.95	1.00	0.98	41
SwipeLeft	1.00	0.95	0.97	41
SwipeRight	0.95	1.00	0.98	42
Accuracy			0.98	207
Macro avg	0.98	0.98	0.98	207
Weighted avg	0.98	0.98	0.98	207

TABLE 18 – Rapport de classification de l'HistGradientBoosting avec filtre

Classe réelle	Classe prédictive	Nombre d'erreurs	Pourcentage d'erreurs (%)
InOut	Click	1	2.38
InOut	Still	2	4.76
SwipeLeft	SwipeRight	2	4.88

TABLE 19 – Tableau des erreurs de classification

5.4 Méthode : Perceptron Multi-Couche

Hyperparamètres utilisés :

- ◊ `dropout_rate = 0,1`
- ◊ `learning_rate = 0,001.`
- ◊ `neurones = 16 – 8.`
- ◊ `epochs = 50.`
- ◊ `batch_size = 16.`

5.4.1 Sans filtre

5.4.1.1 Résultats numériques

Hyperparamètres	Precision (%)	Tps inférence par geste. (ms)
Oui	90.34	1.257

TABLE 20 – Performances

5.4.1.2 Résultats visuels

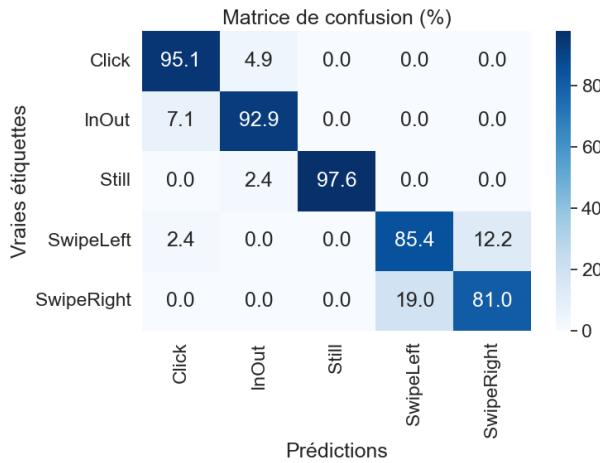


FIGURE 32 – Matrice de confusion sans filtre

Classe	Précision	Rappel	F1-score	Support
Click	0.91	0.95	0.93	41
InOut	0.93	0.93	0.93	42
Still	1.00	0.98	0.99	41
SwipeLeft	0.81	0.85	0.83	41
SwipeRight	0.87	0.81	0.84	42
Accuracy			0.90	207
Macro avg	0.90	0.90	0.90	207
Weighted avg	0.90	0.90	0.90	207

TABLE 21 – Rapport de classification sans filtre

Classe réelle	Classe prédictive	Nombre d'erreurs	Pourcentage d'erreurs (%)
Click	InOut	2	4.88
InOut	Click	3	7.14
Still	InOut	1	2.44
SwipeLeft	Click	1	2.44
SwipeLeft	SwipeRight	5	12.20
SwipeRight	SwipeLeft	8	19.05

TABLE 22 – Tableau des erreurs de classification sans filtre

5.4.2 Avec filtre

Types de filtre appliqués :

- **Filtre passe-bas** : ordre 3, fréquence de coupure $f_c = 0,5$ Hz
- **Savitzky-Golay** : ordre 3, fenêtre $N = 7$
- **EMA (Exponential Moving Average)** : fenêtre $N = 9$

5.4.2.1 Résultats numériques

Filtrage	Hyperparamètres	Précision (%)	Tps d'inférence (ms/geste)
Pass-bas	Oui	92.75	0.116
Savitzky-Golay	Oui	90.34	1.200
EMA	Oui	96.62	1.553

TABLE 23 – Performances avec filtre et réglage d'hyperparamètres

5.4.2.2 Résultats visuels (filtrage EMA avec hyperparamètres)

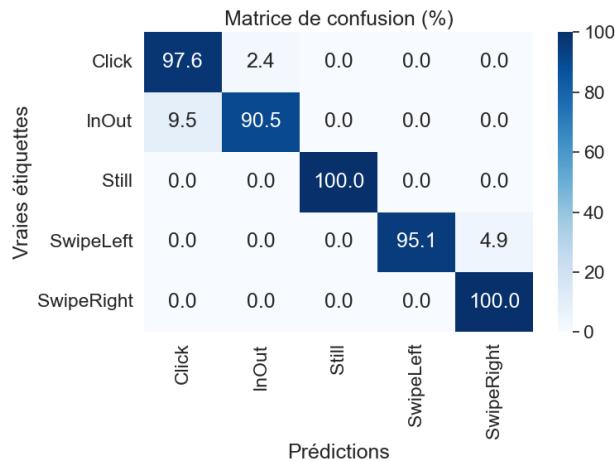


FIGURE 33 – Matrice de confusion avec filtrage

Classe	Précision	Rappel	F1-score	Support
Click	0.91	0.98	0.94	41
InOut	0.97	0.90	0.94	42
Still	1.00	1.00	1.00	41
SwipeLeft	1.00	0.95	0.97	41
SwipeRight	0.95	1.00	0.98	42
Accuracy			0.97	207
Macro avg	0.97	0.97	0.97	207
Weighted avg	0.97	0.97	0.97	207

TABLE 24 – Rapport de classification

Classe réelle	Classe prédite	Nombre d'erreurs	Pourcentage d'erreurs (%)
Click	InOut	1	2.44
InOut	Click	4	9.52
SwipeLeft	SwipeRight	2	4.88

TABLE 25 – Tableau des erreurs de classification

6 Discussion

Dans cette partie, nous faisons le point sur les résultats obtenus et les leçons majeures tirées de ce travail.

6.1 Analyse des résultats

Les résultats mettent en évidence plusieurs tendances marquées. Pour commencer, le **tri des données** occupe une place primordiale. Sans filtrage, des modèles tels que le Random Forest (88%) ou le KNN (84%) rencontrent des difficultés à identifier certains gestes, en particulier les mouvements opposés *SwipeLeft* et *SwipeRight*. Avec la mise en place d'une filtration, notamment à travers le filtre **EMA**, ces confusions sont considérablement diminuées, ce qui permet d'atteindre une précision allant jusqu'à 90%.

L'**HistGradientBoosting (HGB)** se distingue nettement lorsque l'on compare les modèles. Il parvient déjà à 93% sans filtrage et monte jusqu'à 98% avec EMA, tout en maintenant une rapidité d'inférence. Le **Random Forest**, qui est plus simple, a atteint un taux de précision de 97% avec EMA et se révèle donc particulièrement adapté aux applications embarquées en temps réel. Le **Perceptron Multi-Couche (MLP)** affiche également d'excellentes performances (97% avec EMA), cependant, son coût computationnel plus élevé restreint son attrait dans des contextes limités. Au final, le **KNN**, malgré son amélioration par le biais du filtrage, demeure le moins efficace et le plus sensible au bruit.

En ce qui concerne les erreurs, les gestes statiques tels que *Click*, *Still*, *InOut* sont parfaitement identifiés (>95% après filtrage). Les problèmes majeurs demeurent liés aux mouvements dynamiques opposés, cependant les erreurs de confusion deviennent marginales avec l'EMA (<5%).

Pour résumer, pour un équilibre optimal entre **exactitude et célérité**, l'association du Random Forest avec le filtrage EMA s'avère être une option solide. Pour une **exactitude optimale**, l'HGB demeure l'option la plus efficace. Le MLP représente une option alternative valable mais onéreuse, alors que le KNN semble peu approprié.

6.2 Analyse globale des performances

L'évaluation comparative (voir ici) confirme que tous les modèles dépassent 80% de précision, ce qui valide la faisabilité de la reconnaissance de gestes. Leurs comportements diffèrent toutefois en termes de compromis entre précision et rapidité :

- **KNN** : environ **80,7%** de précision avec filtrage et une latence très faible (**0,041 ms/geste**). Idéal pour la réactivité immédiate, mais limité en précision.
- **Random Forest (RF)** : plus de **90%** de précision avec EMA et une latence extrêmement faible (**0,019 ms/geste**). Bon compromis entre rapidité, précision et robustesse.
- **HistGradientBoost (HGB)** : le plus précis (**90%**), mais plus long à entraîner et légèrement plus lent à l'inférence.
- **MLP** : atteint **89,9%**, mais avec une latence plus élevée et un entraînement moins stable. Puissant, mais coûteux en ressources.

Ces résultats illustrent le compromis classique entre **performance brute** (HGB) et **efficacité computationnelle** (KNN, RF). Le MLP se situe entre les deux mais reste gourmand en ressources.

6.3 Influence du filtrage du signal

Le filtrage a été très avantageux pour la plupart des modèles. Il minimise le bruit des signaux bruts et optimise la stabilité des prévisions. Les modèles d'ensemble, notamment, en tirent grand avantage : le filtrage fonctionne comme un lissage qui aide à différencier les gestes similaires.

En pratique, un filtrage approprié permet de :

- diminuer la variabilité entre actions semblables,
- optimiser la stabilité temporelle des prédictions,
- accroître légèrement la précision générale.

Toutefois, un filtrage excessif pourrait supprimer des détails précieux (en particulier dans les composants haute fréquence). Cela met en évidence la nécessité de trouver un équilibre approprié et propose une direction de recherche future : l'exploration de stratégies de filtrage plus détaillées et spécifiques à chaque catégorie de mouvement.

6.4 Impact des hyperparamètres

L'ajustement des hyperparamètres influence directement les performances des modèles. Chaque algorithme a des paramètres qui affectent sa performance en matière d'apprentissage efficace, tout en prévenant le surapprentissage ou, au contraire, un manque d'ajustement.

- ◊ Dans le cas du **MLP**, des paramètres tels que la *profondeur du réseau* sont cruciaux. Un réseau excessivement complexe court le risque de retenir le bruit, alors qu'un réseau trop simple n'exploite pas pleinement les données.
- ◊ Pour les modèles basés sur des arbres (**RF** et **HGB**), la *quantité d'arbres*, la *profondeur maximale* et les méthodes de *régularisation* influencent directement l'exactitude et la durée du calcul. Des configurations trop permissives font grimper le coût sans bénéfice tangible.
- ◊ Pour le **KNN**, la variable cruciale est le nombre de voisins k . Un k trop bas rend le modèle vulnérable au bruit, alors qu'un k trop élevé provoque une sur-généralisation.

Par conséquent, l'absence d'ajustement précis restreint les performances. Une optimisation plus avancée (cross-validation, recherche par grille ou techniques bayésiennes) pourrait accroître la précision tout en contrôlant le temps de calcul.

6.5 Performances computationnelles

L'évaluation des durées d'entraînement et de la latence d'inférence est cruciale pour des applications concrètes, particulièrement en temps réel.

- ◊ Le **KNN** présente l'avantage d'un coût d'apprentissage pratiquement inexistant et d'une latence minimale. Il est donc approprié pour les systèmes embarqués ou interactifs.
- ◊ Le **RF** propose aussi une solution équilibrée, combinant vitesse et solidité, ce qui le rend judicieux pour LinLED.
- ◊ Le **HGB**, qui est plus cher, demeure utilisable sur des systèmes ayant des ressources adéquates. Sa reconnaissance exceptionnelle compense la durée prolongée de son entraînement.
- ◊ Le **MLP** a la latence la plus haute, ce qui restreint son application pour des interactions en temps réel. Il reste néanmoins pertinent si un léger décalage est acceptable.

6.6 Applications en temps réel

Pour des usages pratiques tels que les interfaces utilisateur et les systèmes intégrés, trois aspects sont primordiaux : **latence réduite**, **robustesse** et **consommation modeste en ressources**.

En ce sens, les modèles **KNN** et **Random Forest** semblent être les plus appropriés : ils sont rapides, précis et peu gourmands en ressources, assurant une interaction sans accrocs et sans latence perceptible.

Le **HGB**, qui se distingue par sa haute précision, peut être déployé dans des systèmes munis de ressources matérielles plus robustes.

Bien que le **MLP** soit performant, il présente une latence trop importante pour les situations les plus interactives. Il demeure cependant utilisable dans des situations où une réponse instantanée n'est pas cruciale.

6.6.1 Similitudes entre gestes et rôle de l'interface

L'analyse des matrices de confusion montre que certains gestes proches sont fréquemment confondus, notamment avec le KNN. Cela souligne l'importance d'une interface utilisateur claire et intuitive.

Des pistes d'amélioration incluent :

- ◊ définir des gestes bien distincts pour limiter les ambiguïtés,
- ◊ fournir un *feedback* visuel ou haptique pour guider l'utilisateur,
- ◊ enrichir la base d'entraînement pour mieux différencier les gestes similaires.

6.6.2 Limites

Malgré des résultats encourageants, plusieurs limites méritent d'être soulignées :

- **Base de données limitée** : peu de participants et de répétitions, ce qui réduit la généralisation.
- **Variabilité inter-individuelle** : les différences de style ou d'amplitude peuvent introduire des biais que certains modèles gèrent mal.
- **Choix des métriques** : l'accuracy et la latence sont insuffisantes pour évaluer les gestes rares ou ambigus. Des métriques comme la F1-score seraient plus adaptées.
- **Contraintes matérielles** : les performances de HGB et MLP dépendent fortement du matériel, et pourraient chuter sur des dispositifs embarqués plus limités.

Ces limites appellent à une validation plus large, avec davantage de participants et des conditions variées, pour confirmer la robustesse des modèles.

7 Contribution

Durant de ce stage, j'ai occupé un rôle à la fois polyvalent et central, intervenant sur les volets techniques, expérimentaux et organisationnels. Cette expérience m'a offert l'opportunité de mettre en pratique mes connaissances acquises au cours de ma formation et de les enrichir à travers des missions concrètes et variées.

Dans un premier temps, j'ai mené une recherche bibliographique approfondie sur les interfaces gestuelles, les capteurs infrarouges linéaires et les méthodes d'apprentissage automatique. Cette phase m'a permis de comprendre les différentes approches existantes, d'identifier leurs avantages et limites, et de sélectionner les solutions les plus adaptées à notre projet. Cette étape de réflexion théorique a constitué le socle nécessaire pour orienter les choix techniques et expérimentaux qui ont suivi.

Ensuite, j'ai pris en charge l'assemblage et la calibration du dispositif LinLED. J'ai défini un protocole d'acquisition rigoureux visant à garantir la qualité et la reproductibilité des données, un élément crucial pour la fiabilité des expériences et pour l'entraînement des modèles d'apprentissage automatique. Cette étape a exigé une grande précision et une attention particulière aux détails, car toute erreur dans la capture des signaux pouvait compromettre l'ensemble du projet.

Côté logiciel, j'ai développé les programmes Arduino pour l'acquisition haute vitesse des données et mis en place des outils sous Matlab pour la visualisation et l'enregistrement en temps réel. J'ai également automatisé le prétraitement et la structuration des données, en appliquant différentes méthodes de filtrage et de segmentation afin de réduire le bruit tout en conservant les informations essentielles. Cette étape a été indispensable pour préparer les données et les rendre exploitable par les modèles de classification.

Par la suite, j'ai travaillé sur la conception, l'entraînement et la validation de plusieurs modèles de machine learning, tels que Random Forest, HistGradientBoosting, MLP et KNN, en optimisant leurs hyperparamètres pour obtenir des performances robustes et fiables. J'ai également exploré la mise en oeuvre de ces modèles sur microcontrôleur, afin d'évaluer leur faisabilité et leur réactivité dans un contexte embarqué (XGBoost, Random Forest, ML, ...) avec des contraintes strictes de calcul et de mémoire. Cette partie du projet m'a permis de relier les aspects théoriques de l'apprentissage automatique à des applications concrètes et contraintes par le matériel.

Enfin, j'ai contribué à la dimension organisationnelle et communicationnelle du projet. J'ai participé activement aux réunions et aux séminaires, présenté le projet et les résultats au comité OpenLab Automotive Motion Lab, rédigé le rapport final et centralisé toutes les ressources dans un dépôt GitHub. Ces actions ont permis d'assurer la traçabilité, le partage et la valorisation du travail réalisé.

Au travers de ces différentes missions, j'ai pu développer une expertise complète, allant de l'électronique embarquée au traitement du signal et au machine learning. Ce projet m'a également offert la satisfaction de contribuer concrètement à l'amélioration et à la validation d'un système innovant de reconnaissance gestuelle basé sur LinLED, confirmant ainsi le potentiel de cette technologie pour de futures applications industrielles.

8 Conclusion

Ce projet de stage a permis de concevoir et d'évaluer un système de reconnaissance gestuelle basé sur la technologie **LinLED**, combinant capteurs infrarouges linéaires et algorithmes d'apprentissage automatique. Les objectifs fixés ont été atteints :

- ◊ **Collecte et traitement des données** : une base de cinq gestes (*Click, Swipe Left/Right, Still, In-Out*) a été constituée auprès de 20 participants. Le prétraitement des signaux, notamment grâce à différents filtres, a permis de réduire le bruit tout en conservant les caractéristiques utiles à la classification.
- ◊ **Modélisation et classification** : plusieurs modèles ont été comparés. L'**Histogramme Gradient Boosting** a obtenu les meilleurs scores de précision, tandis que le **Random Forest**, plus léger et rapide, s'est montré particulièrement adapté aux contraintes d'un déploiement embarqué.
- ◊ **Optimisation pour l'embarqué** : les temps d'inférence mesurés (inférieurs à 0,2 ms) assurent une réactivité compatible avec des applications temps réel, en particulier dans le domaine automobile.

Perspectives

Plusieurs pistes d'amélioration peuvent être envisagées :

- ◊ **Extension de la base** : inclure davantage de participants et de gestes plus variés pour améliorer la généralisation des modèles.
- ◊ **Déploiement embarqué** : tester directement le système sur microcontrôleur et en conditions réelles.
- ◊ **Interface interactive** : ajouter un retour visuel ou haptique pour guider l'utilisateur et standardiser les gestes.
- ◊ **Application automobile** : valider l'intégration dans un prototype fonctionnel afin d'évaluer ses performances dans un environnement concret.

Bilan et applications potentielles

Les résultats obtenus ont montré la faisabilité d'une interface gestuelle réactive, précise et peu consommatrice en ressources, adaptée à des environnements contraints comme l'automobile. L'association de LinLED et d'un modèle léger tel que le Random Forest a constitué un compromis optimal entre performance, latence et efficacité énergétique.

Enfin, ce stage m'a offert l'opportunité d'appliquer et de consolider mes compétences en **traitement du signal**, en **apprentissage automatique** et en **systèmes embarqués**. Il m'a également permis de confirmer le potentiel de LinLED comme une solution innovante et prometteuse pour de futures applications industrielles dans le domaine des interfaces gestuelles.

Références

- [1] Stephane VIOLET, Chauvet MARTIN et Ingangiola JEAN-MARC. “LinLED : Low latency and accurate contactless gesture interaction”. In : *Companion Publication of the 25th International Conference on Multimodal Interaction*. 2023, p. 61-65. URL : <https://doi.org/10.1145/3610661.3617164>.
- [2] Kyun Kyu KIM et al. “A substrate-less nanomesh receptor with meta-learning for rapid hand task recognition”. In : *Nature Electronics* 6.1 (2023), p. 64-75. URL : <https://www.nature.com/articles/s41928-022-00888-7>.
- [3] Nibhrat LOHIA et al. “AirWare : Utilizing Embedded Audio and Infrared Signals for In-Air Hand-Gesture Recognition”. In : *arXiv preprint arXiv:2101.10245* (2021). URL : <https://arxiv.org/abs/2101.10245>.
- [4] Anudalem MAEREG et al. “Hand gesture recognition based on near-infrared sensing wristband”. In : (2019). URL : <https://hira.hope.ac.uk/id/eprint/2979/>.
- [5] Tomás MANTECÓN et al. “A real-time gesture recognition system using near-infrared imagery”. In : *PloS one* 14.10 (2019), e0223320. URL : <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0223320>.
- [6] Keunwoo PARK et al. “DeepFisheye : Near-surface multi-finger tracking technology using fisheye camera”. In : *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 2020, p. 1132-1146. URL : <https://dl.acm.org/doi/abs/10.1145/3379337.3415818>.
- [7] Yuki YAMATO et al. “Hand gesture interaction with a low-resolution infrared image sensor on an inner wrist”. In : *Proceedings of the 2020 International Conference on Advanced Visual Interfaces*. 2020, p. 1-5. URL : <https://dl.acm.org/doi/abs/10.1145/3399715.3399858>.
- [8] Fahmid AL FARID et al. “A structured and methodological review on vision-based hand gesture recognition system”. In : *Journal of Imaging* 8.6 (2022), p. 153. URL : <https://www.mdpi.com/2313-433X/8/6/153>.
- [9] Shuo JIANG et al. “Emerging wearable interfaces and algorithms for hand gesture recognition : A survey”. In : *IEEE Reviews in Biomedical Engineering* 15 (2021), p. 85-102. URL : <https://ieeexplore.ieee.org/abstract/document/9426433>.
- [10] Walter HEILIGENBERG et Joseph BASTIAN. “The electric sense of weakly electric fish.” In : *Annual review of physiology* 46 (1984), p. 561-583. URL : <https://europepmc.org/article/med/6324664>.
- [11] Gerald WESTHEIMER et Suzanne P MCKEE. “Spatial configurations for visual hyperacuity”. In : *Vision research* 17.8 (1977), p. 941-947. URL : <https://www.sciencedirect.com/science/article/abs/pii/0042698977900694>.
- [12] Frédéric LEGRAND et Licence Creative COMMONS. “Échantillonnage et reconstruction d'un signal périodique”. In : (). URL : <https://www.f-legrand.fr/scidoc/srcdoc/numerique/tfd/echantillonnage/echantillonnage-pdf.pdf>.
- [13] Radwan JALAM. “Apprentissage automatique et catégorisation de textes multilingues”. Thèse de doct. Lyon 2, 2003. URL : <https://theses.fr/2003LY020056>.
- [14] Hasan Ahmed SALMAN, Ali KALAKECH et Amani STEITI. “Random forest algorithm overview”. In : *Babylonian Journal of Machine Learning* 2024 (2024), p. 69-79. URL : <https://journals.mesopotamian.press/index.php/BJML/article/view/417>.
- [15] Stéphane CARON. “Une introduction aux arbres de décision”. In : *Récupéré sur* (2011). URL : <https://scaron.info/doc/intro-arbres-decision/intro.pdf>.
- [16] Zhongheng ZHANG. “Introduction to machine learning : k-nearest neighbors”. In : *Annals of translational medicine* 4.11 (2016), p. 218. URL : <https://pmc.ncbi.nlm.nih.gov/articles/PMC4916348/>.
- [17] Li YANG et Abdallah SHAMI. “On hyperparameter optimization of machine learning algorithms : Theory and practice”. In : *Neurocomputing* 415 (2020), p. 295-316. URL : <https://www.sciencedirect.com/science/article/abs/pii/S0925231220311693>.

- [18] Mohammad MAFTOUN et al. “Malicious URL Detection using optimized Hist Gradient Boosting Classifier based on grid search method”. In : *arXiv preprint arXiv* :2406.10286 (2024). URL : <https://arxiv.org/abs/2406.10286>.
- [19] Sankar K PAL et Sushmita MITRA. “Multilayer perceptron, fuzzy sets, and classification”. In : *IEEE Transactions on neural networks* 3.5 (1992), p. 683-697.
- [20] Marc PARIZEAU. “Réseaux de neurones”. In : *GIF-21140 et GIF-64326* 124 (2004). URL : http://frostiebek.free.fr/docs/Reseaux%20de%20neuronnes/RNF_fuzzy.pdf.
- [21] Navid Fazle RABBI. “Design, Implementation, Comparison, and Performance analysis between Analog Butterworth and Chebyshev-I Low Pass Filter Using Approximation, Python and Proteus”. In : *arXiv preprint arXiv* :2102.09048 (2021). URL : <https://arxiv.org/abs/2102.09048>.
- [22] Koray S ERER. “Adaptive usage of the Butterworth digital filter”. In : *Journal of biomechanics* 40.13 (2007), p. 2934-2943. URL : <https://www.sciencedirect.com/science/article/abs/pii/S0021929007001029>.
- [23] David C STONE. “Application of median filtering to noisy data”. In : *Canadian Journal of chemistry* 73.10 (1995), p. 1573-1581. URL : <https://cdnsciencepub.com/doi/abs/10.1139/v95-195>.
- [24] Neal B GALLAGHER. “Savitzky-golay smoothing and differentiation filter”. In : *Eigenvector Research Incorporated* 2 (2020), p. 4. URL : <https://eigenvector.com/wp-content/uploads/2020/01/SavitzkyGolay.pdf>.
- [25] Jason BROWNLEE. *Deep learning for time series forecasting : predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.
- [26] Niels BANTILAN. “pandera : Statistical Data Validation of Pandas Dataframes.” In : *SciPy*. 2020, p. 116-124. URL : <https://pdfs.semanticscholar.org/99b3/57333595e7ab6c21d6fc63bb63f0e3bdb7b0.pdf>.

Table des matières

1	Introduction	4
1.1	Contexte du projet	4
1.2	Présentation de l’Institut des Sciences du Mouvement (ISM)	4
1.3	Objectifs du projet	5
1.4	Problématique	5
1.5	Organisation du rapport	6
2	État de l’art	7
3	Matériels	9
3.1	Maquette d’acquisition	9
3.1.1	Le dispositif LinLED	9
3.1.2	Principe de fonctionnement de LinLED	10
3.1.3	Acquisition des signaux et connexion avec la Teensy 4.1	12
3.2	Configuration machine	13
3.2.1	Matériel utilisé	13
3.2.2	Ordinateur utilisé	14
3.2.3	Environnement logiciel	14
4	Méthodes	15
4.1	Création et mise en forme de la base de données	15
4.1.1	Base de données	16
4.2	Apprentissage automatique	20
4.3	Filtres	25
4.3.1	Filtre Passe-Bas	26
4.3.2	Filtre de Savitzky-Golay	26
4.3.3	Filtre EMA (Exponential Moving Average)	27
4.4	Segmentation des gestes	28
4.4.1	Principe de fonctionnement	28
4.4.2	Sortie de la fonction	29
4.4.3	Intérêt de l’extraction	29
4.5	Extraction des caractéristiques des gestes	29
4.5.1	Principe de la méthode	29
4.5.2	Sortie de la fonction	30
4.5.3	Intérêt de l’extraction	30

5 Résultats	31
5.1 Méthode : Random Forest	31
5.1.1 Sans filtre	31
5.1.2 Avec filtre	32
5.2 Méthode : KNN	33
5.2.1 Sans filtre	34
5.2.2 Avec filtre	35
5.3 Méthode : HistGradientBoosting	36
5.3.1 Sans filtre	36
5.3.2 Avec filtre	37
5.4 Méthode : Perceptron Multi-Couche	39
5.4.1 Sans filtre	39
5.4.2 Avec filtre	40
6 Discussion	42
6.1 Analyse des résultats	42
6.2 Analyse globale des performances	42
6.3 Influence du filtre du signal	42
6.4 Impact des hyperparamètres	43
6.5 Performances computationnelles	43
6.6 Applications en temps réel	43
6.6.1 Similitudes entre gestes et rôle de l'interface	44
6.6.2 Limites	44
7 Contribution	45
8 Conclusion	46

Table des figures

2	Exemple d'interface sans contact intégrée dans un tableau de bord.	4
3	Système LinLED accompagné d'un retour visuel par LED.	5
4	Dispositif LinLED	9
6	Illustrations du prototype LinLED	10
7	Poisson électrique	10
8	Illustration des gaussiennes pondérées (Source)	11
9	Disposition des LEDs et photodiodes (Source)	11
10	Brochage de la sortie analogique (Analog OUTPUT)	12
11	Présentation de la carte Teensy 4.1	13
12	Exemple de signal analogique et numérique, illustrant la conversion du signal continu en valeurs discrètes. Source	15
13	Signal issu des 16 canaux (capteurs) pendant un mouvement de la main	18
17	Schéma du processus d'apprentissage automatique (Source)	21
18	Principe de fonctionnement d'un classifieur Random Forest (Source)	22
19	Illustration du principe de KNN (Source)	23
20	Fonctionnement d'un modèle de type Boosting (Source)	23
21	Représentation simplifiée d'un réseau de neurones. (Source)	24
22	Effet du filtre passe-bas sur la somme pondérée du signal du geste <i>Still</i>	26
23	Effet du filtre Savitzky-Golay sur la somme pondérée du signal du geste <i>Still</i>	27
24	Effet du filtre EMA sur la somme pondérée du signal du geste <i>Still</i>	28
25	Chaîne de traitement : de l'acquisition à l'apprentissage	30
26	Matrice de confusion de Random Forest sans filtrage	31
27	Matrice de confusion de Random Forest avec filtrage	33
28	Matrice de confusion de KNN sans filtrage	34
29	Matrice de confusion de KNN avec filtrage	35
30	Matrice de confusion de l'HistGradientBoosting sans filtrage	37
31	Matrice de confusion de l'HistGradientBoosting avec filtrage	38
32	Matrice de confusion sans filtrage	39
33	Matrice de confusion avec filtrage	41

Liste des tableaux

1	Affectation des broches du connecteur de sortie analogique	12
2	Performances de Random Forest sans filtrage	31
3	Rapport de classification de Random Forest sans filtrage	32
4	Tableau des erreurs de classification de Random Forest sans filtrage	32
5	Performances de Random Forest avec filtrage	32
6	Rapport de classification de Random Forest avec filtrage	33
7	Tableau des erreurs de classification de Random Forest avec filtrage	33
8	Performances de KNN sans filtrage	34
9	Rapport de classification de KNN sans filtrage	34
10	Tableau des erreurs de classification de KNN sans filtrage	35
11	Performances de KNN avec filtrage	35
12	Rapport de classification de KNN avec filtrage	36
13	Tableau des erreurs de classification de KNN avec filtrage	36
14	Performances de l'HistGradientBoosting sans filtrage	36
15	Rapport de classification de l'HistGradientBoosting sans filtrage	37
16	Tableau des erreurs de classification de l'HistGradientBoosting sans filtrage	37
17	Performances avec filtrage et réglage d'hyperparamètres	38
18	Rapport de classification de l'HistGradientBoosting avec filtrage	38
19	Tableau des erreurs de classification	38
20	Performances	39
21	Rapport de classification sans filtrage	40
22	Tableau des erreurs de classification sans filtrage	40
23	Performances avec filtrage et réglage d'hyperparamètres	40
24	Rapport de classification	41
25	Tableau des erreurs de classification	41
26	Résultats selon le filtrage passe-bas et le réglage manuel des hyperparamètres	53

ANNEXES

Performances obtenues

Afin d'évaluer l'effet du réglage manuel des hyperparamètres et du filtrage passe-bas sur la performance du modèle Random Forest, nous présentons dans le tableau ci-dessous les résultats obtenus pour différentes configurations.

Les tirets ” - ” indiquent que le paramètre n'a pas été utilisé.

n_est	max_d	min_split	min_leaf	seed	acc (%)	tps d'inf (ms/geste)	ord	freq (Hz)
-	-	-	-	42	88.41	0.137	-	-
50	None	2	1	42	88.41	0.082	-	-
100	None	2	1	42	88.41	0.145	-	-
50	5	2	1	42	84.06	0.062	-	-
50	None	5	1	42	88.89	0.063	-	-
50	None	10	1	42	86.96	0.029	-	-
50	None	5	2	42	88.41	0.066	-	-
-	-	-	-	42	85.99	0.076	2	0.1
-	-	-	-	42	86.96	0.076	2	0.5
-	-	-	-	42	89.37	0.101	2	0.9
-	-	-	-	42	85.99	0.081	3	0.1
-	-	-	-	42	86.47	0.124	3	0.5
-	-	-	-	42	90.34	0.105	3	0.9
-	-	-	-	42	87.44	0.082	4	0.1
-	-	-	-	42	88.41	0.098	4	0.5
-	-	-	-	42	86.96	0.103	4	0.9
-	-	-	-	42	86.96	0.096	5	0.1
-	-	-	-	42	87.44	0.020	5	0.5
-	-	-	-	42	87.92	0.100	5	0.9
50	None	2	1	42	89.37	0.031	3	0.9
50	5	2	1	42	84.06	0.047	3	0.9
50	None	5	1	42	90.34	0.040	3	0.9
50	None	2	2	42	87.92	0.019	3	0.9
100	None	2	1	42	90.34	0.058	3	0.9
.100	5	2	1	42	84.06	0.049	3	0.9
100	None	5	1	42	88.41	0.077	3	0.9
100	None	2	2	42	88.89	0.122	3	0.9

TABLE 26 – Résultats selon le filtrage passe-bas et le réglage manuel des hyperparamètres

Les abréviations utilisées dans le tableau sont les suivantes :

- **n_est** : nombre d'arbres dans le modèle (n_estimators)
- **max_d** : profondeur maximale des arbres (max_depth)
- **min_split** : nombre minimal d'échantillons pour diviser un noeud (min_samples_split)
- **min_leaf** : nombre minimal d'échantillons dans une feuille (min_samples_leaf)
- **acc** : précision du modèle en pourcentage (Accuracy %)
- **tps d'inf** : temps d'inférence par geste en millisecondes (Latence ms/geste)
- **ord** : ordre du filtre passe-bas appliqué
- **freq** : fréquence du filtre passe-bas en hertz (Hz)