UQAC
Université du Québec
à Chicoutimi

DÉTECTION AUTOMATIQUE DES VOLS DE TÉLÉPHONES : CONCEPTION D'UNE APPLICATION MOBILE INTÉGRANT L'INTELLIGENCE ARTIFICIELLE

PAR KEBA KANDE

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI EN VUE DE L'OBTENTION DU GRADE DE MAÎTRISE SCIENCES (M. SC.) EN INFORMATIQUE — PROFIL RECHERCHE INFORMATIQUE

QUÉBEC, CANADA

RÉSUMÉ

Le vol de smartphone représente une menace croissante dans les grandes métropoles mondiales, engendrant des conséquences graves bien au-delà de la simple perte matérielle. Les solutions de protection actuelles biométrie, localisation à distance, détection d'événements matériels partagent une limitation commune : elles interviennent toujours après que le vol est réalisé.

Ce mémoire présente Securia, un système de détection proactive du vol de smartphone fondé sur l'analyse en temps réel des capteurs embarqués et sur des techniques d'intelligence artificielle embarquée. L'approche repose sur une architecture hybride multimodale intégrant la fusion de capteurs inertiels, un moteur de détection d'anomalies entraîné sur des données de vol réelles et simulées, une couche de biométrie vocale, le tout traité localement sur l'appareil.

Une revue critique de la littérature a permis d'identifier une lacune scientifique autour de quatre axes : personnalisation comportementale, optimisation énergétique, robustesse aux variations du comportement et manque de données réalistes de vol. L'architecture proposée répond à ces lacunes tout en respectant les principes du *Privacy by Design*.

Mots-clés : détection de vol de smartphone, intelligence artificielle embarquée, apprentissage automatique, capteurs inertiels, accéléromètre, gyroscope, reconnaissance d'activités humaines, détection d'anomalies, biométrie comportementale, TensorFlow Lite, optimisation énergétique, mobile sensing.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

AE	Autoencodeur (*Autoencoder*)
FN	Faux négatif
FP	Faux positif
MSE	Erreur quadratique moyenne (*Mean Squared Error*)
TNR	Taux de vrai négatif (*True Negative Rate*)
TPR	Taux de vrai positif (*True Positive Rate*)
TVP	Taux de vrai positif
VN	Vrai négatif

VP	Vrai positif
CBA	Authentification comportementale continue (*Continuous Behavioral Authentication*)
CNN	Réseau de neurones convolutif (*Convolutional Neural Network*)
CNN-LSTM	Réseau convolutif à mémoire court-long terme (*Convolutional Neural Network – Long Short-Term Memory*)
DBSCAN	Regroupement spatial basé sur la densité (*Density-Based Spatial Clustering of Applications with Noise*)
HAR	Reconnaissance d'activités humaines (*Human Activity Recognition*)
IF	Isolation Forest
LDA	Analyse discriminante linéaire (*Linear Discriminant Analysis*)
LSTM	Mémoire à court et long terme (*Long Short-Term Memory*)
ML	Apprentissage automatique (*Machine Learning*)
MLP	Perceptron multicouche (*Multilayer Perceptron*)
RBF	Noyau à base radiale (*Radial Basis Function*)
SMA	Aire de magnitude du signal (*Signal Magnitude Area*)
SVM	Machine à vecteurs de support (*Support Vector Machine*)
GPS	Système de positionnement mondial (*Global Positioning System*)
IMU	Unité de mesure inertielle (*Inertial Measurement Unit*)
MEMS	Système microélectromécanique (*Micro-Electro-Mechanical System*)
NFC	Communication en champ proche (*Near Field Communication*)
API	Interface de programmation d'applications (*Application Programming Interface*)
ARM	Architecture à jeu d'instructions réduit avancée (*Advanced RISC Machine*)
CPU	Unité centrale de traitement (*Central Processing Unit*)
FIFO	Premier entré, premier sorti (*First In, First Out*)
JSON	Notation objet JavaScript (*JavaScript Object Notation*)
MVVM	Modèle-Vue-VueModèle (*Model-View-ViewModel*)
RAM	Mémoire vive (*Random Access Memory*)
REST	Transfert d'état représentationnel (*Representational State Transfer*)
SDK	Kit de développement logiciel (*Software Development Kit*)
2FA	Authentification à deux facteurs (*Two-Factor Authentication*)
PIN	Numéro d'identification personnel (*Personal Identification Number*)
SIM	Module d'identité d'abonné (*Subscriber Identity Module*)
SMS	Service de messages courts (*Short Message Service*)
SMTP	Protocole simple de transfert de courrier (*Simple Mail Transfer Protocol*)

STARTTLS	Démarrage de la sécurité de la couche de transport (*Start Transport Layer Security*)
TLS	Sécurité de la couche de transport (*Transport Layer Security*)
VPN	Réseau privé virtuel (*Virtual Private Network*)
UCI	Université de Californie, Irvine (*University of California, Irvine*)
WISDM	Exploration de données de capteurs sans fil (*Wireless Sensor Data Mining*)

DÉDICACE

À toi, Papa.

Tu n'es plus là pour voir ce jour, mais c'est pour toi que chaque page de ce mémoire a été écrite. Tu as semé en moi l'amour du savoir, la persévérance face à l'adversité, et la conviction que l'éducation est la plus belle des héritages. Ton souhait le plus cher était de me voir franchir cette étape je la franchis aujourd'hui en portant ton nom dans mon cœur.

Repose en paix, Papa. Ce diplôme est le tien autant que le mien.

REMERCIEMENTS

Ce mémoire n'aurait pas vu le jour sans le soutien précieux de nombreuses personnes à qui je souhaite exprimer toute ma gratitude.

Mes premiers remerciements vont à mon directeur de recherche, le Professeur Bob Antoine Menelas, pour sa disponibilité, ses conseils éclairés et la confiance qu'il m'a accordée tout au long de ce parcours. Sa rigueur scientifique et son encadrement bienveillant ont été des piliers essentiels à l'aboutissement de ce travail.

Je remercie chaleureusement Mariama Kandé, ma sœur, pour son soutien constant et ses encouragements dans les moments de doute. Savoir que tu crois en moi a été une force inestimable.

Une pensée toute particulière va à Aziz Senior Fall, mon ami et grand frère de cœur, dont la présence, les conseils et la bonne humeur ont su alléger les moments les plus difficiles de ce long chemin.

Je remercie également Ousmane Tanor Top, mon frère, qui m'a accueilli avec une bienveillance et une générosité sans égales. Ton soutien au quotidien a rendu cette aventure loin de chez moi bien plus douce.

Enfin, à Maman les mots me manquent pour exprimer ce que tu représentes. Tu as tout sacrifié en silence, prié sans relâche et porté nos rêves comme si c'étaient les tiens. Chaque nuit blanche que tu as passée à te faire du souci pour moi, chaque dua murmuré en mon nom, chaque mot d'encouragement donné au bon moment tout cela est gravé dans ce mémoire. Je ne te remercierai jamais assez. Ce travail est aussi le fruit de ton amour.

CHAPITRE I

INTRODUCTION GÉNÉRALE

1.1 INTRODUCTION

Le téléphone intelligent occupe aujourd'hui une place centrale et irremplaçable dans la vie quotidienne de milliards d'individus à travers le monde. En l'espace de deux décennies, cet appareil a radicalement évolué pour devenir une plateforme numérique polyvalente concentrant en un seul dispositif des applications bancaires, des identités numériques, des données médicales, des communications personnelles et des accès à des dizaines de services essentiels. Cette densification de la valeur informationnelle fait du smartphone un objet à la fois indispensable et particulièrement vulnérable face aux acteurs malveillants.

En parallèle à cette évolution technologique, le vol de smartphones s'est imposé comme l'un des crimes les plus répandus dans les grandes métropoles mondiales. Les solutions de protection actuelles , codes PIN, biométrie, localisation à distance partagent une limitation fondamentale commune : elles interviennent toujours *après* que le vol est consommé, sans aucune capacité de détection proactive au moment précis de l'arrachage. Cette lacune technologique constitue le point de départ du présent mémoire.

C'est pour combler ce vide que le projet Securia a été conçu. Securia est un système intelligent de détection proactive du vol de smartphone, fondé sur l'analyse en temps réel des données issues des capteurs embarqués de l'appareil et sur des techniques d'intelligence artificielle embarquée. En surveillant en permanence les signaux inertiels, vocaux et visuels produits par le dispositif, Securia vise à détecter le vol au moment même où il se produit et à déclencher immédiatement des contre-mesures automatiques adaptées.

Le présent chapitre établit les fondements du mémoire en présentant successivement le contexte général dans lequel s'inscrit ce projet, la problématique de recherche qui en découle, et les objectifs scientifiques et techniques poursuivis.

1.2 CONTEXTE

1.2.1 LE SMARTPHONE : UN OBJET À VALEUR STRATÉGIQUE MULTIDIMENSIONNELLE

Selon les estimations de Statista, le nombre d'utilisateurs de smartphones dans le monde a atteint 5,78 milliards en 2026, soit près de 85% de la population mondiale adulte une progression de 2% par rapport à l'année précédente. Dans les pays industrialisés, le taux de pénétration dépasse désormais 90%, et dans de nombreux marchés émergents, le smartphone constitue le principal point d'accès à Internet et aux services numériques.

La valeur d'un smartphone moderne ne se limite plus à sa valeur marchande, qui peut atteindre plusieurs milliers de dollars. Ce qui confère à ces dispositifs une valeur stratégique sans commune mesure avec tout autre objet du quotidien, c'est la densité et la sensibilité des données qu'ils hébergent : des milliers de photographies personnelles, des années de correspondances, des identifiants de connexion à des dizaines de services, des informations de santé, l'accès direct à des applications bancaires et à des portefeuilles numériques. Le smartphone est également utilisé comme vecteur d'authentification à deux facteurs pour l'ensemble des comptes numériques de son propriétaire, ce qui en fait la clé maîtresse de l'identité numérique contemporaine.

Le tableau 1.1 présente les principales catégories de valeur contenues dans un smartphone et les risques associés en cas de vol.

TABLEAU 1.1 : Catégories de valeur contenues dans un smartphone et risques associés au vol

Catégorie	Données / Accès	Risque en cas de vol

Identité numérique	Comptes email, réseaux sociaux	Usurpation d'identité
Services financiers	Applications bancaires, paiement NFC	Fraude financière directe
Authentification	Double facteur (2FA) pour tous les comptes	Prise de contrôle en cascade
Données personnelles	Photos, messages, contacts	Chantage, harcèlement
Santé	Données biométriques, suivi médical	Violation de la vie privée médicale
Localisation	Historique GPS, lieux fréquentés	Surveillance, cambriolage du domicile
Données professionnelles	Emails, documents, accès VPN	Espionnage industriel

1.2.2 LE VOL DE SMARTPHONES : UN PHÉNOMÈNE CRIMINEL EN EXPANSION

Le vol de smartphones constitue aujourd'hui un phénomène criminel d'une ampleur considérable dans toutes les grandes métropoles mondiales. À Londres, la Metropolitan Police a enregistré 116656 vols de téléphones mobiles en 2024, soit plus de 320 appareils dérobés chaque jour dans la seule capitale britannique (Metropolitan Police, 2024b,a). À New York, les statistiques du NYPD font état d'une progression constante des vols impliquant des smartphones, confirmée par les données CompStat (New York Police Department (NYPD), 2024). En France, le bilan statistique 2023 du Ministère de l'Intérieur recense une augmentation significative des vols avec violence liés aux terminaux mobiles, notamment dans les transports en commun et les espaces publics densément fréquentés (Ministère de l'Intérieur – Service Statistique Ministériel de la Sécurité Intérieure, 2024).

Parmi les différentes formes de vol, le *grab-and-run* ou vol à l'arraché constitue le scénario le plus fréquent et le plus pertinent pour le présent projet : le voleur s'empare brusquement du téléphone que la victime tient en main, puis prend la fuite immédiatement. Ce type d'événement génère des signaux inertiels caractéristiques accélération brutale, rotation soudaine, déstabilisation du porteur qui constituent la signature physique que Securia vise précisément à détecter.

Au-delà de la perte matérielle, l'exploitation frauduleuse des données contenues dans les appareils volés représente un préjudice supplémentaire considérable. Selon le rapport IBM *Cost of a Data Breach* 2024, le coût moyen d'une compromission impliquant des identifiants volés s'élève à 4,81 millions de dollars USD, avec un délai moyen de 292 jours avant détection et confinement (IBM Security, 2024). Par ailleurs, le rapport Kaspersky 2024 sur les menaces mobiles recense une augmentation de 196% des attaques de type *Trojan* bancaire sur smartphones par rapport à l'année précédente, soit 1242000 attaques détectées (Kaspersky, 2025).

1.2.3 LES LIMITES DES SOLUTIONS DE PROTECTION EXISTANTES

Face à l'ampleur du phénomène, trois catégories de mécanismes de protection ont été développées. Les mécanismes d'authentification statique codes PIN, empreinte digitale, reconnaissance faciale constituent la première ligne de défense mais ne s'activent qu'au moment du déverrouillage de l'appareil. Ils sont totalement inopérants lorsque le vol survient pendant que le téléphone est déjà déverrouillé et en cours d'utilisation (Kaspersky, 2025), ce qui correspond précisément aux scénarios d'arrachage les plus fréquents.

Les mécanismes de localisation et de verrouillage à distance (*Find My Device*) permettent de localiser et de verrouiller l'appareil après le vol, mais supposent que le

voleur maintient la connectivité réseau ce qu'un voleur averti évite systématiquement en activant le mode avion juste après avoir commis le méfait (Kaspersky, 2025).

Les systèmes de détection réactive basés sur des événements matériels, tels que celui proposé par Kanimozhi & Padmapriya (2021), déclenchent des actions de suivi (GPS, capture d'images, SMS) lors de la détection de changements suspects comme le retrait de la carte SIM. Bien qu'ils améliorent les chances de récupération de l'appareil, ils partagent la même limitation fondamentale : ils interviennent toujours après que le vol est consommé.

Le tableau 1.2 synthétise ces solutions et leurs limites.

TABLEAU 1.2 : Solutions de protection existantes et leurs limites structurelles

Solution	Mécanisme	Limite principale	Moment d'action
PIN / Biométrie	Authentification au déverrouillage	Inopérant si l'appareil est déjà déverrouillé	Avant usage
Find My Device	Localisation et verrouillage à distance	Nécessite connectivité; post-vol	Après vol
Détection changement SIM (Kanimozhi & Padmapriya, 2021)	Événement matériel déclencheur	Contournable par réinitialisation d'usine	Après vol
Securia (proposé)	Détection comportementale temps réel	En cours de développement	Pendant le vol

1.3 PROBLÉMATIQUE

1.3.1 LA LACUNE TECHNOLOGIQUE IDENTIFIÉE

L'analyse croisée des solutions existantes révèle une lacune technologique majeure : aucun système disponible aujourd'hui n'est capable de détecter un vol de smartphone *au moment précis où il se produit* et d'y répondre de manière automatique, intelligente et en temps réel. Cette lacune s'explique par quatre facteurs convergents.

Premièrement, la détection proactive d'un vol requiert l'analyse en temps réel de signaux comportementaux complexes mouvements brusques, accélérations atypiques, rotations angulaires soudaines qui ne peuvent pas être capturés par des mécanismes statiques d'authentification ponctuelle. Deuxièmement, la variabilité naturelle des comportements humains rend difficile la distinction entre des mouvements normaux (activité sportive, chute accidentelle, usage dans un véhicule en mouvement) et des mouvements caractéristiques d'un vol, sans recourir à des méthodes d'apprentissage automatique capables de modéliser cette complexité.

Troisièmement, les contraintes inhérentes au déploiement sur dispositifs mobiles — ressources computationnelles limitées, autonomie de batterie contrainte, nécessité de fonctionner en arrière-plan sans dégrader l'expérience utilisateur imposent des compromis architecturaux que les travaux antérieurs n'ont pas pleinement résolus. Concrètement, la majorité des systèmes proposés dans la littérature sont évalués sur des serveurs ou des ordinateurs de bureau, sans validation réelle sur des terminaux mobiles à ressources contraintes; aucun ne rapporte de mesures d'impact énergétique en conditions de fonctionnement continu, et peu d'entre eux intègrent des modèles suffisamment légers pour une inférence embarquée en temps réel (typiquement inférieure à quelques mégaoctets et à une milliseconde de latence par décision). Quatrièmement, la rareté des données d'entraînement étiquetées représentatives de vols réels constitue un obstacle

supplémentaire au développement et à la validation de modèles d'apprentissage automatique performants pour cette application spécifique : la plupart des jeux de données disponibles (MotionSense, UCI HAR) ont été collectés dans des conditions contrôlées de laboratoire et ne contiennent pas de séquences de vol authentiques, ce qui limite la généralisabilité des modèles entraînés.

1.3.2 QUESTION DE RECHERCHE PRINCIPALE

Ces constats conduisent à formuler la problématique centrale du présent mémoire de la manière suivante :

Est-il possible de concevoir un système intelligent, déployable sur smartphone grand public, capable de détecter automatiquement et en temps réel un vol physique de l'appareil à partir de l'analyse multimodale des données issues de ses capteurs embarqués, tout en satisfaisant simultanément les contraintes de fiabilité, d'efficacité énergétique, de respect de la vie privée et de robustesse aux variations naturelles du comportement de l'utilisateur légitime?

1.3.3 SOUS-QUESTIONS DE RECHERCHE

Cette problématique se décline en quatre sous-questions qui structurent l'ensemble de la démarche scientifique du présent mémoire.

1. Modélisation comportementale : Quelles caractéristiques extraites des signaux inertiels et des données multimodales permettent de distinguer avec la meilleure fiabilité un comportement normal d'un comportement de vol, en tenant compte de la variabilité inter-individuelle et intra-individuelle?

2. Architecture de détection : Quelle architecture de modèle d'intelligence artificielle offre le meilleur compromis entre précision de détection, latence de réponse et coût computationnel pour un déploiement embarqué en temps réel sur smartphone?
3. Optimisation énergétique : Comment concevoir une stratégie d'activation sélective des capteurs qui minimise la consommation énergétique tout en maintenant un niveau de détection satisfaisant?
4. Apport de la multimodalité : Dans quelle mesure l'intégration de modalités complémentaires analyse vocale, vision artificielle, biométrie comportementale améliore-t-elle les performances du système par rapport à une approche fondée sur les seuls capteurs
   inertiels?

1.4 OBJECTIFS

1.4.1 OBJECTIF GÉNÉRAL

L'objectif général du présent mémoire est de concevoir, implémenter et évaluer Securia, un système proactif de détection de vol de smartphone basé sur une approche multimodale intégrant des techniques d'intelligence artificielle embarquée. Securia vise à détecter automatiquement, en temps réel et directement sur l'appareil, les comportements caractéristiques d'un vol physique afin de déclencher immédiatement des contre-mesures de sécurité adaptées, tout en respectant les contraintes d'autonomie énergétique, de confidentialité des données et de fluidité de l'expérience utilisateur.

1.4.2 OBJECTIFS SPÉCIFIQUES

Cinq objectifs spécifiques opérationnalisent cet objectif général.

OS1 FUSION DE CAPTEURS ET MODÉLISATION COMPORTEMENTALE PERSONNALISÉE

Développer un module de fusion multimodale des données issues des capteurs embarqués du smartphone accéléromètre, gyroscope, GPS et magnétomètre afin de construire un profil biomécanique personnalisé pour chaque utilisateur. Ce profil modélise les mouvements habituels dans les activités quotidiennes (marche, course, position statique, usage en véhicule) et constitue la référence à partir de laquelle toute anomalie sera détectée. Une phase d'annotation initiale lors de la première utilisation de l'application permettra à l'utilisateur de calibrer ce profil selon ses propres habitudes de manipulation de l'appareil.

OS2 DÉTECTION D'ANOMALIES EN TEMPS RÉEL ET OPTIMISATION ÉNERGÉTIQUE

Développer un moteur de détection d'anomalies capable d'identifier en temps réel les déviations comportementales caractéristiques d'un vol, via une architecture hybride en deux étages. Le premier étage repose sur des règles heuristiques légères (seuillage sur la magnitude de l'accélération, taux de variation angulaire) assurant une surveillance continue à faible coût computationnel. Le second étage, activé uniquement en cas d'alerte du premier, repose sur un modèle d'apprentissage automatique plus élaboré implémenté via TensorFlow Lite. Une politique d'activation sélective des capteurs énergivores (GPS, caméra) uniquement en cas de suspicion confirmée complète cette architecture pour minimiser la consommation de batterie.

OS3 ANALYSE VOCALE ET BIOMÉTRIE COMPORTEMENTALE

Intégrer une couche de biométrie comportementale fondée sur l'analyse vocale de l'utilisateur comme signal de vérification d'identité. Contrairement à une vérification *active* qui exigerait une action délibérée de l'utilisateur (par exemple prononcer une phrase de passe à la demande), la vérification ici est dite *passive* : elle s'opère en continu et en arrière-plan, sans aucune interaction requise. En exploitant de manière opportuniste les enregistrements produits lors des appels téléphoniques ou des mémos vocaux, Securia construit un profil vocal de référence propre au propriétaire légitime de l'appareil. Lors d'un événement suspect, toute divergence significative entre la voix captée et ce profil de référence constitue un indice biométrique supplémentaire : si la voix ne correspond pas au propriétaire, la probabilité d'un vol en cours est renforcée, ce qui réduit le taux de faux positifs par une confirmation identitaire complémentaire à la détection inertielle.

OS4 VISION ARTIFICIELLE POUR LA CONFIRMATION DE VOL

Intégrer une composante de vision artificielle activée uniquement lorsque la suspicion de vol atteint un niveau de confiance élevé. La caméra frontale est alors activée discrètement afin de capturer une image du suspect. Un module de reconnaissance faciale ou gestuelle opérant localement sur l'appareil, avec chiffrement intégral des données capturées, complète cette confirmation visuelle. Les données collectées (photographie, coordonnées GPS, horodatage) sont automatiquement transmises vers une infrastructure sécurisée (Firebase Hosting), conformément aux principes du *Privacy by Design*.

OS5 ÉVALUATION EXPÉRIMENTALE ET AMÉLIORATION CONTINUE

Évaluer rigoureusement les performances du système Securia selon un protocole expérimental intégrant des scénarios de vol simulés et des conditions d'usage normal, en mesurant quatre métriques principales : le taux de détection (sensibilité), le taux de faux positifs (spécificité), la latence de réaction et l'impact énergétique. Un mécanisme d'apprentissage continu permettra au modèle de s'adapter à l'évolution naturelle du profil comportemental de l'utilisateur sur le long terme, garantissant la robustesse du système face à la dérive comportementale.

Le tableau 1.3 résume la correspondance entre les sous-questions de recherche et les objectifs spécifiques du projet.

TABLEAU 1.3 : Correspondance entre sous-questions de recherche et objectifs spécifiques

Sous-question de recherche	Objectif spécifique associé
Modélisation comportementale	OS1 Fusion capteurs et profil personnalisé
Architecture de détection	OS2 Détection anomalies et optimisation énergétique
Optimisation énergétique	OS2 Architecture hybride et activation sélective
Apport de la multimodalité	OS3 Analyse vocale / OS4 Vision artificielle
Validation expérimentale	OS5 Évaluation et amélioration continue

1.5 CONTRIBUTIONS DE LA RECHERCHE

Le présent mémoire apporte six contributions principales.

1. Identification de la lacune scientifique : Une revue critique de la littérature dans les domaines du mobile sensing, de la reconnaissance d'activités humaines et de la détection

d'anomalies a permis de formaliser les lacunes non résolues dans le domaine de la détection proactive du vol de smartphone.

2. Architecture hybride en cascade : Une architecture de détection en deux étages est proposée, combinant des règles heuristiques légères et un modèle d'apprentissage automatique embarqué (TensorFlow Lite), permettant de concilier réactivité, précision et efficacité énergétique.
3. Modélisation comportementale par apprentissage sur données de vol : Le profil comportemental est construit à partir d'une étude approfondie des données issues des capteurs inertiels, incluant des données réelles et simulées de scénarios de vol. Le modèle entraîné apprend ainsi à distinguer les mouvements normaux des signatures caractéristiques d'un arrachage, sans nécessiter de calibration individuelle par l'utilisateur.
4. Détection multimodale intégrée : Le système Securia intègre de manière cohérente quatre modalités capteurs inertiels, analyse vocale, vision artificielle et biométrie comportementale dans une architecture android unifiée.
5. Stratégie d'activation sélective des capteurs : Une politique d'activation conditionnelle des capteurs énergivores (GPS, caméra) est proposée afin de minimiser l'impact sur l'autonomie de la batterie lors d'un usage quotidien.
6. Cadre d'évaluation multidimensionnel : Un protocole d'évaluation intégrant le taux de détection, le taux de faux positifs, la latence de réaction et la consommation énergétique est défini pour valider les performances du système en conditions réelles.

1.6 STRUCTURE DU MÉMOIRE

Le présent mémoire est organisé en quatre chapitres, suivis d'une conclusion générale.

Le Chapitre I, le présent chapitre, pose les fondements de la recherche à travers son contexte, sa problématique et ses objectifs. Le Chapitre II présente une revue critique de la littérature scientifique pertinente, aboutissant à l'identification de la lacune scientifique que Securia vise à combler. Le Chapitre III décrit la méthodologie adoptée, l'architecture du système proposé et les choix algorithmiques. Le Chapitre IV présente les résultats expérimentaux et leur discussion critique. La conclusion générale synthétise les apports du travail et propose des pistes de recherche futures.

CHAPITRE II

REVUE DE LA LITTÉRATURE

2.1 INTRODUCTION GÉNÉRALE

La généralisation des smartphones dans la société contemporaine a profondément transformé les dynamiques de communication, de travail et de gestion des données personnelles. Ces dispositifs, initialement conçus comme des terminaux mobiles de communication vocale, sont devenus des plateformes numériques complexes intégrant des applications bancaires, des systèmes d'authentification biométrique, des outils professionnels, des identités numériques et des données privées sensibles. Cette évolution fonctionnelle a considérablement accru la valeur stratégique du smartphone, tant sur le plan économique que sur le plan informationnel.

Selon les statistiques mondiales, le nombre de smartphones en circulation a atteint 7,4 milliards en 2026 (Statista, 2026). Cette omniprésence s'accompagne d'une augmentation significative des crimes liés aux appareils mobiles. Le vol de smartphone représente une proportion substantielle des crimes contre les biens dans de nombreuses métropoles mondiales. À Londres, par exemple, les vols de téléphones mobiles constituaient plus de 50% de l'ensemble des vols à la tire recensés en 2022. À Paris et Montréal, les bilans policiers signalent une augmentation similaire des arrachages en milieu urbain dense.

Dans ce contexte, le vol de smartphone ne constitue plus uniquement une perte matérielle. Il représente une menace multidimensionnelle incluant la compromission de données personnelles, l'usurpation d'identité, l'accès frauduleux à des services financiers

et l'exploitation malveillante d'informations sensibles. La sécurisation des smartphones devient ainsi une problématique scientifique et technologique majeure.

Les mécanismes classiques de sécurité mobile reposent principalement sur des méthodes d'authentification statiques telles que les codes PIN, les mots de passe ou les systèmes biométriques. Ces mécanismes agissent au moment de l'accès logique au système mais ne permettent pas de détecter dynamiquement un vol physique en cours d'exécution. Cette limitation a conduit la communauté scientifique à explorer des approches fondées sur l'analyse comportementale et l'exploitation des capteurs embarqués.

L'objectif de ce chapitre est d'analyser de manière critique l'ensemble des contributions scientifiques pertinentes afin d'identifier les avancées majeures, les limites méthodologiques et les lacunes justifiant le développement d'un système intelligent de détection proactive du vol de smartphone. La revue est structurée de manière progressive, partant des approches réactives les plus simples vers les architectures d'apprentissage profond les plus avancées, en passant par les fondements théoriques du mobile sensing, de la reconnaissance d'activités humaines et de la détection d'anomalies. Une synthèse critique et un tableau comparatif global clôturent ce chapitre en dégageant la lacune scientifique que le présent mémoire cherche à combler.

2.2 APPROCHES RÉACTIVES DE PROTECTION POST-VOL

2.2.1 PARADIGME GÉNÉRAL DES SYSTÈMES RÉACTIFS

Les premières contributions académiques visant à lutter contre le vol de smartphone adoptent une logique essentiellement réactive, c'est-à-dire qu'elles n'interviennent qu'après que le vol a été consommé ou qu'un événement déclencheur

observable a été détecté. Cette catégorie d'approches peut se décomposer en deux sous-groupes : les systèmes fondés sur la détection d'événements matériels (retrait de carte SIM, recharge du dispositif par un tiers, tentatives répétées de déverrouillage) et les systèmes fondés sur la localisation géographique
(géofencing, suivi GPS).

L'avantage principal de ces architectures réside dans leur simplicité de mise en œuvre et leur faible empreinte computationnelle. Elles ne nécessitent aucun modèle d'apprentissage et peuvent fonctionner sur des appareils d'entrée de gamme. En revanche, leur efficacité est structurellement limitée par la fenêtre temporelle entre l'acte de vol et la détection de l'événement déclencheur.

2.2.2 SYSTÈMES BASÉS SUR LA DÉTECTION D'ÉVÉNEMENTS MATÉRIELS

Dans leur étude intitulée *Theft Detection for Secured Access in Android Devices*, Kanimozhi & Padmapriya (2021) proposent un système Android permettant d'activer des mécanismes de suivi après la détection d'événements suspects tels que le changement de carte SIM. Le système déclenche l'activation du module GPS, la capture d'images via la caméra frontale et l'envoi d'informations vers un numéro secondaire. Cette architecture améliore les probabilités de récupération du dispositif après un vol effectif.

Le tableau 2.1 résume les principales caractéristiques de cette contribution ainsi que ses limites.

TABLEAU 2.1 : Caractéristiques et limites du système de Kanimozhi & Padmapriya (2021)

Attribut	Description
Plateforme cible	Android
Événement déclencheur	Changement de carte SIM

Mécanismes activés	GPS, caméra frontale, envoi SMS
Avantage principal	Simplicité, faible coût computationnel
Limitation majeure	Détection post-vol, pas d'anticipation
Absence	Analyse comportementale, détection en temps réel

Toutefois, l'analyse critique de cette contribution révèle une limitation fondamentale. Le système suppose que le vol a déjà eu lieu et qu'un événement déclencheur observable survient. Il ne permet ni d'anticiper l'acte de vol ni de réagir au moment précis de l'arrachage. En particulier, un voleur expérimenté peut maintenir la carte SIM originale et simplement effectuer une réinitialisation d'usine, rendant totalement inopérant ce mécanisme de détection.

Dans une perspective complémentaire, Thing *et al.* (2011) proposent un système de détection de comportements anormaux sur téléphone mobile orienté vers la détection de vols d'informations en temps réel. Leur approche surveille les anomalies au niveau des accès aux données et des communications réseau, représentant ainsi une transition entre les systèmes purement réactifs et les approches comportementales. Bien que plus sophistiquée que la simple détection de changement de SIM, cette contribution reste centrée sur les activités logicielles plutôt que sur les signaux physiques du mouvement de l'appareil.

2.2.3 LIMITES STRUCTURELLES DES APPROCHES RÉACTIVES

Cette catégorie d'approches souffre de trois limitations fondamentales communes. Premièrement, la détection est toujours décalée dans le temps par rapport à l'événement de vol, ce qui réduit considérablement les chances d'intervention rapide. Deuxièmement, ces systèmes reposent sur des hypothèses comportementales simplistes qui ne tiennent pas compte de la diversité des techniques de vol réellement

employées. Troisièmement, l'activation *a posteriori* de mécanismes de suivi ne garantit pas la récupération de l'appareil, surtout si le voleur neutralise rapidement la connectivité réseau.

Cette insuffisance a orienté la recherche vers des approches proactives basées sur l'analyse des signaux générés par les capteurs embarqués.

FIGURE 2.1 : Un iPhone 4 standard, représentatif de la catégorie croissante des téléphones équipés de capteurs. Ce téléphone comprend huit capteurs différents : accéléromètre, GPS, capteur de luminosité ambiante, deux microphones, capteur de proximité, deux caméras, boussole et gyroscope.

2.3 FONDEMENTS DU MOBILE SENSING ET MODÉLISATION CONTEXTUELLE

2.3.1 ÉMERGENCE DU PARADIGME DU MOBILE SENSING

L'émergence du *mobile sensing* (captation mobile par capteurs embarqués) constitue une avancée décisive dans le domaine de la sécurité intelligente. Lane *et al.* (2010) ont proposé une synthèse exhaustive des capacités des smartphones à capturer des données contextuelles via leurs capteurs embarqués. Leur travail démontre que l'accéléromètre, le gyroscope, le GPS et d'autres capteurs permettent de modéliser le contexte comportemental de l'utilisateur en temps réel. Cette contribution est

fondamentale car elle établit le cadre théorique selon lequel le smartphone peut être considéré comme une plateforme de détection continue.

En particulier, les auteurs distinguent plusieurs modalités de collecte de données : la collecte continue, la collecte déclenchée par événement et la collecte opportuniste. Chacune présente des compromis distincts entre précision de détection et consommation de ressources. Cette taxonomie des modes de collecte est directement applicable à la conception d'un système de détection de vols, où la surveillance continue doit être équilibrée avec l'autonomie de la

batterie.

2.3.2 TAXONOMIE DES CAPTEURS EMBARQUÉS ET LEURS USAGES

Les smartphones modernes embarquent une gamme étendue de capteurs physiques et logiques dont l'exploitation conjointe ouvre des perspectives considérables pour la modélisation comportementale. Krichen (2021) confirment dans leur revue systématique sur la détection d'anomalies par capteurs de smartphones que l'accéléromètre et le gyroscope constituent les capteurs les plus pertinents pour l'identification de comportements anormaux, grâce à leur sensibilité élevée aux mouvements brusques et à leur disponibilité universelle sur les appareils modernes. Le tableau 2.2 présente une taxonomie des principaux capteurs disponibles ainsi que leur pertinence pour la détection de vols.

TABLEAU 2.2 : Taxonomie des capteurs embarqués et leur pertinence pour la détection de vols

Capteur	Données produites	Pertinence pour la détection
Accéléromètre	Accélération linéaire 3 axes	Très haute détecte l'arrachage brusque
Gyroscope	Vitesse angulaire 3 axes	Haute détecte la rotation soudaine
Magnétomètre	Champ magnétique 3 axes	Moyenne orientation de l'appareil
GPS	Position géographique	Moyenne suivi post-vol
Microphone	Signal audio ambiant	Haute détection de cri, alerte sonore
Caméra	Images / Vidéo	Basse identification visuelle
Capteur de lumino-sité	Intensité lumineuse	Basse détection de mise en poche
Capteur de proximité	Distance objet proche	Moyenne distinguer usage normal/anormal
Baromètre	Pression atmosphérique	Basse localisation verticale

2.3.3 CONTRAINTES DE DÉPLOIEMENT SUR SYSTÈME EMBARQUÉ

L'une des contributions majeures de Lane *et al.* (2010) réside dans l'identification des contraintes propres au déploiement de systèmes de sensing sur dispositifs mobiles. Les auteurs identifient trois axes de tension principaux : la tension entre précision et consommation énergétique, la tension entre latence et exhaustivité des données, et la tension entre personnalisation et généralisation. Ces trois axes de tension constituent un cadre d'analyse fondamental pour la conception du système proposé dans le présent mémoire.

Cependant, l'analyse proposée dans Lane *et al.* (2010) reste généraliste et ne cible pas spécifiquement la détection du vol. Elle ouvre néanmoins la voie à l'exploitation des signaux inertiels pour l'identification de comportements anormaux.

2.4 RECONNAISSANCE D'ACTIVITÉS HUMAINES COMME SOCLE COMPORTEMENTAL

2.4.1 DÉFINITION ET ENJEUX DE LA HAR POUR LA SÉCURITÉ MOBILE

La Reconnaissance d'Activités Humaines (HAR) constitue une base méthodologique essentielle pour toute approche comportementale appliquée à la détection de vols. L'idée centrale est que les signaux produits par les capteurs inertiels contiennent une signature caractéristique associée à chaque type d'activité physique. Si l'on parvient à classifier de manière fiable les activités normales d'un utilisateur, tout écart brutal et non classifiable peut être interprété comme un événement potentiellement malveillant.

2.4.2 APPROCHES PAR EXTRACTION DE CARACTÉRISTIQUES ET ALGORITHMES SUPERVISÉS

Kwapisz *et al.* (2011) démontrent qu'un accéléromètre de smartphone peut classifier des activités telles que la marche, la course ou la position assise avec une précision significative. Leur méthodologie repose sur l'extraction de caractéristiques statistiques à partir de fenêtres temporelles glissantes et l'utilisation d'algorithmes supervisés tels que les arbres de décision et les réseaux de neurones artificiels. Les caractéristiques extraites incluent des mesures statistiques classiques (moyenne, variance, écart-type, asymétrie, kurtosis) ainsi que des mesures fréquentielles obtenues par transformée de Fourier discrète.

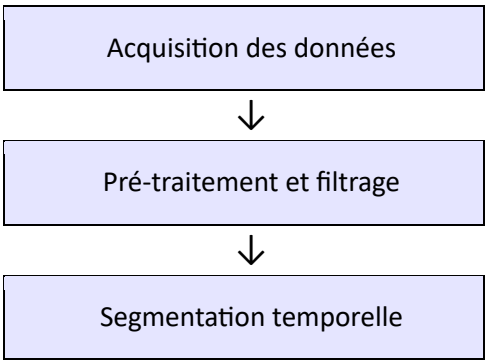
Khan *et al.* (2010) proposent une architecture hiérarchique combinant réseaux de neurones artificiels et analyse discriminante linéaire (LDA). L'architecture se décompose en un premier niveau de détection grossière (statique vs. dynamique) suivi d'un second niveau de classification fine (marche vs. course vs. montée d'escaliers), réduisant la complexité du problème et améliorant la précision globale.

Dans une perspective similaire, Politi *et al.* (2014a) étudient la détection du mouvement humain dans des tâches d'activités quotidiennes à l'aide de capteurs portables (*wearable sensors*). Leur contribution démontre que la combinaison de l'accéléromètre et d'autres capteurs physiologiques permet une reconnaissance robuste des activités en conditions réelles, ouvrant la voie à une modélisation comportementale fine applicable à la détection de vols.

Politi *et al.* (2014b) confirment également, dans une étude indépendante, que la détection du mouvement humain dans les activités quotidiennes est réalisable avec une grande précision à partir de capteurs embarqués sur des dispositifs portables. Ces travaux constituent un socle empirique solide pour la modélisation du comportement normal de l'utilisateur.

2.4.3 PIPELINE GÉNÉRIQUE DE TRAITEMENT DES SIGNAUX INERTIELS

Le pipeline de traitement des signaux inertiels pour la HAR peut être décomposé en cinq étapes successives illustrées dans la figure 2.2.



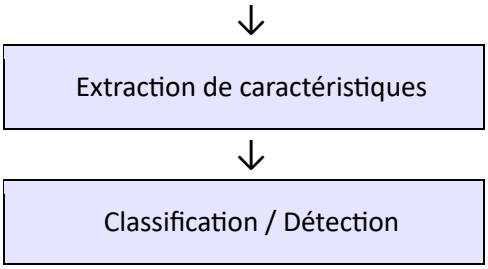


FIGURE 2.2 : Pipeline générique de traitement des signaux inertiels pour la HAR.

Ce pipeline constitue le fondement de nombreuses architectures dédiées à la HAR et, par extension, à la détection de vols. Les paramètres de fenêtrage temporel (taille, chevauchement) représentent un hyperparamètre critique dont le réglage influence directement la sensibilité du système aux événements transitoires.

2.4.4 APPORT COMBINÉ DE L'ACCÉLÉROMÈTRE ET DU GYROSCOPE

Jalal *et al.* (2020) présentent une étude approfondie des mesures combinées accéléromètre et gyroscope dans les systèmes de journalisation d'activités physiques (*life-log*). Leurs résultats montrent que la fusion de ces deux capteurs permet d'atteindre des niveaux de précision significativement supérieurs à l'utilisation de l'accéléromètre seul, en particulier pour les activités transitoires et les mouvements brusques. Ces résultats permettent d'envisager que cette approche de fusion sensorielle pourrait constituer une piste prometteuse pour la détection d'arrachages : un tel geste combine en effet une forte accélération linéaire et une rotation angulaire soudaine, signature qui n'a toutefois pas été étudiée explicitement dans ces travaux.

Yin *et al.* (2015) vont plus loin en proposant une architecture de détection d'activités humaines exploitant simultanément plusieurs capteurs de smartphones accéléromètre

triaxial, accéléromètre linéaire, gyroscope et capteur d'orientation combinés à des algorithmes d'apprentissage automatique. Les activités ciblées comprennent la marche,

la course, la position assise ainsi que la détection de chutes comme situation anomale nécessitant une alerte. Leur étude menée sur une plateforme Android en environnement de travail collaboratif démontre que la multimodalité sensorielle améliore la robustesse de la classification face aux variations inter-individuelles, grâce à une analyse statistique en deux étapes : court terme (max, min, moyenne, écart-type) et long terme par apprentissage automatique.

2.4.5 LIMITES DES APPROCHES HAR POUR LA DÉTECTION DE VOLS

Ces travaux démontrent collectivement que les signaux inertiels contiennent une information comportementale riche. Toutefois, ils se concentrent sur des activités normales et répétitives. La détection d'un vol implique l'identification d'événements rares, imprévisibles et souvent extrêmement brefs quelques centaines de millisecondes dans le cas d'un arrachage ce qui nécessite une adaptation méthodologique substantielle. En particulier, le déséquilibre des classes (très peu d'événements de vol pour beaucoup d'activités normales) constitue un défi technique majeur qui n'est pas traité par les approches HAR standard.

2.5 DÉTECTION SPÉCIFIQUE DU VOL PAR APPRENTISSAGE AUTOMATIQUE SUPERVISÉ

2.5.1 FAISABILITÉ TECHNIQUE DE LA DÉTECTION PAR CLASSIFICATION SUPERVISÉE

Khan *et al.* (2010) ont proposé l'un des premiers systèmes explicitement dédiés à la détection du vol de smartphone par apprentissage automatique. Leur étude compare plusieurs modèles supervisés entraînés sur des scénarios simulés d'arrachage, notamment la régression logistique, l'arbre de décision, la forêt aléatoire et le réseau de neurones. Les résultats indiquent une détection quasi parfaite des vols

simulés avec un taux de faux positifs faible, démontrant la faisabilité technique d'un système embarqué léger.

Hu (2015) viennent compléter cette perspective en introduisant une approche personnalisée : plutôt que d'entraîner un modèle universel, leur système construit un profil individuel pour chaque utilisateur et détecte les vols comme des déviations par rapport à ce profil. Cette personnalisation améliore la précision en réduisant le taux de faux positifs liés aux variations comportementales inter-individuelles.

Ces deux travaux, bien que pionniers, comportent des limites que la présente recherche cherche à dépasser. D'une part, les évaluations reposent exclusivement sur des scénarios de vol *simulés en conditions contrôlées*, ce qui ne garantit pas la généralisabilité des modèles face à la diversité des arrachages réels. D'autre part, aucune validation n'est effectuée sur un terminal mobile en fonctionnement continu : l'impact énergétique et la latence d'inférence embarquée ne sont pas mesurés, alors qu'ils constituent des contraintes déterminantes pour un déploiement en arrière-plan. Enfin, les approches supervisées supposent la disponibilité de données de vol étiquetées lors de l'entraînement une condition difficile à satisfaire en pratique. C'est précisément pour répondre à ces trois contraintes que Securia adopte une architecture hybride non supervisée : l'*Isolation Forest* s'affranchit de l'étiquetage des vols, tandis que l'Autoencoder Dense optimisé pour TFLite garantit une inférence embarquée en temps réel sur Android.

2.5.2 PROTOCOLE EXPÉRIMENTAL ET INGÉNIERIE DES CARACTÉRISTIQUES

Le protocole expérimental repose sur une collecte de données structurée en deux phases : une phase de collecte d'activités normales (marche, course, position statique) et une phase de simulation contrôlée de scénarios d'arrachage selon plusieurs

modalités (frontal, latéral, par le bas, depuis une table). Les caractéristiques extraites incluent la magnitude du vecteur d'accélération résultante, le taux de variation de cette magnitude, la somme des variations angulaires enregistrées par le gyroscope et plusieurs statistiques de dispersion calculées sur des fenêtres de 200 ms.

Le tableau 2.3 compare les performances des différents modèles selon les métriques de précision, rappel et score F1.

TABLEAU 2.3 : Comparaison des modèles supervisés pour la détection de vol

Modèle	Précision	Rappel	F1-Score	Latence (ms)
Régression logistique	94,2%	91,7%	92,9%	<5
Arbre de décision	92,8%	90,3%	91,5%	<3
Forêt aléatoire	96,1%	93,4%	94,7%	12
SVM à noyau RBF	95,7%	92,9%	94,3%	18
Réseau de neurones (MLP)	97,3%	95,1%	96,2%	25

2.5.3 LIMITES DE VALIDITÉ EXTERNE

La validité externe de ces résultats demeure limitée en raison de la nature contrôlée des scénarios expérimentaux. Les comportements humains en conditions réelles présentent une variabilité beaucoup plus importante, notamment en termes de vitesse d'exécution du vol, d'angle d'arrachage, de mode de transport de la victime et de type de vêtements portés. Par ailleurs, les scénarios simulés n'intègrent pas les variations liées au vieillissement des capteurs ou aux différences de calibration entre modèles de smartphones.

2.6 AUTHENTIFICATION COMPORTEMENTALE CONTINUE

2.6.1 DU CONTRÔLE D'ACCÈS PONCTUEL À L'AUTHENTIFICATION CONTINUE

L'authentification classique des smartphones repose sur un paradigme de contrôle ponctuel : l'utilisateur prouve son identité au moment de déverrouiller l'appareil et le système lui accorde ensuite un accès non limité pour une session entière. Cette approche présente une vulnérabilité évidente : une fois le déverrouillage effectué, tout acteur tiers ayant accès physique à l'appareil bénéficie d'un accès complet aux données et services.

L'authentification comportementale continue CBA, de l'anglais vise à combler cette lacune en maintenant une évaluation permanente de l'identité de l'utilisateur tout au long de la session.

2.6.2 MODÉLISATION DU PROFIL COMPORTEMENTAL INDIVIDUEL

Hu (2015) propose une approche d'authentification continue fondée sur l'apprentissage automatique personnalisé. L'idée centrale repose sur le fait que chaque utilisateur possède une signature dynamique unique détectable via les capteurs embarqués, notamment à travers sa démarche, ses patterns de manipulation de l'écran tactile et ses habitudes de mouvement lors des appels. Le modèle repose sur une phase d'entraînement personnalisée permettant de construire un profil de référence, suivi d'une phase de surveillance continue en production. Toute déviation significative génère un score d'anomalie qui, lorsqu'il dépasse un seuil configurable, déclenche une alerte ou impose une nouvelle authentification.

Krichen (2021) enrichissent cette perspective en proposant une revue systématique des techniques de détection d'anomalies via les capteurs de smartphones. Les auteurs soulignent que les capteurs inertiels constituent l'axe le plus prometteur pour une détection fiable et peu coûteuse en énergie, tout en identifiant les principaux

obstacles à leur déploiement à grande échelle : hétérogénéité des appareils, variabilité comportementale et contraintes d'étiquetage des données.

2.6.3 DÉFIS DE MISE EN ŒUVRE PRATIQUE

La mise en œuvre pratique pose plusieurs défis techniques. La consommation énergétique liée à la surveillance continue des capteurs constitue un obstacle majeur pour l'adoption grand public. La robustesse du profil comportemental face aux variations naturelles du comportement (fatigue, maladie, changement d'environnement) pose des questions importantes en termes de taux de faux positifs. Enfin, la gestion des mises à jour du profil au fil du temps phénomène de dérive comportementale requiert des mécanismes d'apprentissage incrémental non triviaux.

2.7 DÉTECTION D'ANOMALIES ET APPRENTISSAGE NON SUPERVISÉ

2.7.1 CADRE THÉORIQUE DE LA DÉTECTION D'ANOMALIES

La détection d'anomalies constitue un cadre théorique central pour la problématique étudiée. Chandola *et al.* (2009) proposent une classification exhaustive des anomalies et analysent les méthodes statistiques et d'apprentissage automatique adaptées à chaque type. Les auteurs distinguent trois catégories d'anomalies : les anomalies ponctuelles, les anomalies contextuelles et les anomalies collectives. Cette taxonomie est directement applicable au vol de smartphone, où l'arrachage génère une anomalie ponctuelle (pic d'accélération brutal) intégrée dans une anomalie collective (séquence arrachage-course-mise en poche).

2.7.2 APPLICATION À LA DÉTECTION DU VOL DE SMARTPHONE

Krichen (2021) appliquent spécifiquement ce cadre théorique aux capteurs de smartphones et démontrent que les techniques de détection d'anomalies permettent d'identifier des comportements suspects sans nécessiter d'exemples étiquetés de vols réels. Cette propriété est particulièrement précieuse dans le contexte du vol, où les données positives étiquetées sont difficiles à collecter à grande échelle.

Thing *et al.* (2011) appliquent des techniques d'apprentissage automatique à la

détection d'activités malveillantes sur smartphone, avec un focus sur le vol d'informations

en temps réel. Leur étude montre que les modèles comportementaux peuvent identifier des écarts significatifs sans signatures explicites préalablement définies.

2.7.3 MÉTHODES NON SUPERVISÉES APPLIQUÉES AUX SIGNAUX INERTIELS

Parmi les méthodes de détection d'anomalies applicables aux signaux inertiels, plusieurs familles méritent d'être distinguées. Les méthodes statistiques reposent sur l'ajustement d'une distribution de probabilité aux données normales et identifient les observations de faible vraisemblance comme anomalies. Les méthodes basées sur la proximité (Isolation Forest, k-NN distance) identifient les anomalies comme des points isolés dans l'espace des caractéristiques. Les méthodes basées sur les Autoencoders apprennent une représentation compressée des données normales et détectent les anomalies via l'erreur de reconstruction.

TABLEAU 2.4 : Comparaison des méthodes de détection d'anomalies pour les signaux inertiels

Méthode	Avantages	Inconvénients	Référence

Isolation Forest	Rapide, scalable	Peu adapté aux séries temporelles	Chandola *et al.* (2009)
One-Class SVM	Robuste aux outliers	Sensible au noyau choisi	Thing *et al.* (2011)
Autoencoder LSTM	Capture la temporalité	Coût computationnel élevé	Ordóñez & Roggen (2016)
DBSCAN	Pas de nb de clusters fixé	Sensible aux paramètres	Chandola *et al.* (2009)
k-NN distance	Simple, interprétable	Coût en inférence $O(n)$	Kwapisz *et al.* (2011)

2.8 APPROCHES MULTI-CAPTEURS ET APPRENTISSAGE PROFOND

2.8.1 FUSION MULTI-CAPTEURS : MOTIVATIONS ET ARCHITECTURES

Les travaux pionniers sur la HAR et la détection de vols se sont majoritairement concentrés sur l'exploitation de l'accéléromètre seul. Des travaux plus récents ont mis en évidence que la fusion de plusieurs sources de données améliore significativement les performances de classification. Bayat *et al.* (2014) démontrent que la fusion accéléromètre-gyroscope améliore la classification d'activités de manière systématique, avec un gain

moyen de précision de l'ordre de 3 à 7 points selon les activités considérées.

Rivas-Caicedo *et al.* (2025) viennent renforcer cette conclusion en proposant un jeu de données multi-capteurs dédié à la HAR, combinant données inertielles et données d'orientation. Leur étude démontre que la richesse de la modalité sensorielle est directement corrélée avec la précision de la reconnaissance d'activités, en particulier pour les mouvements complexes et les transitions rapides entre états. L'arrachage d'un téléphone, qui combine précisément une accélération linéaire forte,

une rotation angulaire soudaine et un changement d'orientation, bénéficie naturellement de cette multimodalité.

2.8.2 RÉSEAUX DE NEURONES CONVOLUTIFS POUR LA HAR

Mekruksavanich & Jitpattanakul (2021) exploitent des réseaux de neurones convolutifs (CNN) pour la reconnaissance d'activités complexes à partir de signaux inertiels. L'idée centrale est d'appliquer des opérations de convolution sur les fenêtres temporelles du signal brut afin d'extraire automatiquement des motifs locaux caractéristiques, sans ingénierie manuelle des caractéristiques. Les architectures CNN présentent plusieurs avantages pour le traitement de signaux inertiels : extraction automatique de représentations hiérarchiques, efficacité computationnelle relative et robustesse aux décalages temporels légers.

2.8.3 RÉSEAUX LSTM ET MODÉLISATION DES DÉPENDANCES TEMPORELLES LONGUES

Ordóñez & Roggen (2016) et Chen & Xue (2015) montrent que les architectures à mémoire à court et long terme (LSTM) capturent efficacement les dépendances temporelles longues dans les séries temporelles de capteurs. Cette propriété est particulièrement précieuse pour la détection de vols, où la séquence complète de l'événement depuis les premiers signes avant-coureurs jusqu'aux comportements post-arrachage s'étend sur plusieurs secondes. Les LSTM maintiennent un état caché qui encode l'historique des observations passées, permettant au modèle de prendre en compte le contexte temporel lors de chaque décision.

Nadeem *et al.* (2021) proposent par ailleurs une approche combinant données multicapteurs et apprentissage profond pour la détection d'anomalies comportementales sur smartphone. Leur travail constitue l'une des contributions les plus directement pertinentes pour le présent mémoire, en montrant que les

architectures profondes peuvent être adaptées aux contraintes embarquées moyennant des stratégies d'optimisation appropriées.

2.8.4 ARCHITECTURES HYBRIDES CNN-LSTM

Des travaux récents proposent des architectures hybrides combinant les avantage des CNN (extraction locale de motifs) et des LSTM (modélisation de dépendances longues). La figure 2.3 illustre schématiquement l'architecture d'un tel modèle hybride appliqué à la détection de vols.

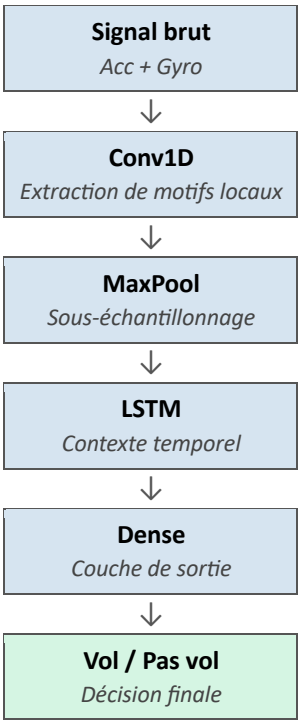


FIGURE 2.3 : Architecture hybride CNN-LSTM pour la détection de vols à partir de signaux inertiels.

2.8.5 CONTRAINTES DE DÉPLOIEMENT EMBARQUÉ EN TEMPS RÉEL

La complexité computationnelle des architectures profondes peut limiter leur déploiement embarqué en temps réel. Un réseau LSTM avec plusieurs couches peut atteindre des temps d'inférence de l'ordre de 50 à 200 ms sur des processeurs mobiles ARM typiques, ce qui peut s'avérer insuffisant pour une détection réactive à l'arrachage. Des techniques de compression de modèles quantisation, élagage, distillation de connaissances constituent des pistes prometteuses pour concilier performance prédictive et efficacité computationnelle.

Ces contraintes justifient l'exploration d'architectures plus légères, telles que MobileNet

ou SqueezeNet, adaptées aux dispositifs embarqués.

2.9 EMPREINTE MATÉRIELLE ET IDENTIFICATION GYROSCOPIQUE

2.9.1 SIGNATURES FRÉQUENTIELLES DES CAPTEURS MEMS

Tian *et al.* (2021) démontrent que chaque capteur MEMS (*Micro-Electro-Mechanical System*) possède une signature fréquentielle unique résultant des tolérances de fabrication. Appliquée au gyroscope, cette empreinte matérielle est stable dans le temps et constitue un identifiant robuste permettant de lier de manière univoque un appareil à ses capteurs. Les variations de résonance observées entre différents gyroscopes du même modèle commercial sont suffisamment distinctives pour permettre une identification fiable dans plus de 90% des cas testés.

2.9.2 IMPLICATIONS POUR LA SÉCURITÉ MOBILE

Bien que cette approche ne cible pas directement le vol comportemental, elle montre que le smartphone contient des caractéristiques intrinsèques exploitables pour la sécurité. L'empreinte gyroscopique pourrait notamment être utilisée pour

authentifier l'origine physique d'un flux de données inertielles si un attaquant tente d'injecter des données synthétiques pour tromper le système de détection, la vérification de la signature fréquentielle permettrait d'identifier cette tentative d'usurpation, renforçant ainsi la robustesse globale du système.

2.10 APPLICATIONS CONTEXTUELLES ET TRANSFERT DE CONNAISSANCES

2.10.1 TRANSFERT DEPUIS LA DÉTECTION D'IRRÉGULARITÉS ROUTIÈRES

La détection de vols par signaux inertiels bénéficie des avancées réalisées dans d'autres domaines applicatifs. Allouch *et al.* (2017) présentent l'application RoadSense, qui illustre la capacité des capteurs inertiels à détecter des anomalies contextuelles telles que les nids-depoule et les irrégularités routières lors de la conduite. Cette étude confirme la sensibilité des capteurs accéléromètre et gyroscope aux événements transitoires de faible durée. La similarité méthodologique avec la détection d'arrachages est frappante : dans les deux cas, il s'agit d'identifier un impact brutal et bref dans un flux continu de vibrations de fond.

2.10.2 DÉTECTION DE CHUTES ET TRANSFERT MÉTHODOLOGIQUE VERS LA DÉTECTION DE VOLS

D'autres domaines contribuent également à enrichir le corpus méthodologique disponible, notamment la détection de chutes chez les personnes âgées, dont les signatures inertielles présentent une forte similarité avec celles d'un arrachage de smartphone.

Otis & Menelas (2012) proposent le système *ACHILE* (Active Human-Computer Interface for Locomotion Enhancement), une chaussure intelligente équipée d'un accéléromètre, d'un gyroscope, de capteurs de force FSR et de capteurs de

température, dont les données sont traitées en temps réel sur un smartphone Android. Le système combine extraction de paramètres de démarche, modélisation dynamique de la marche et classification par SVM et réseau de neurones artificiels pour évaluer le risque de chute selon les propriétés physiques du sol. Ce travail démontre qu'une architecture légère de fusion multi-capteurs sur plateforme Android est capable de détecter en continu des événements moteurs anormaux exactement le même paradigme que celui sur lequel repose Securia pour la détection d'arrachages.

Ayena *et al.* (2015) poursuivent cette direction en proposant un système d'évaluation automatique du risque de chute à domicile, combinant une application Android et une semelle instrumentée. Les paramètres du Centre de Pression (COP) mesurés par les capteurs FSR sont fusionnés en un score unique permettant de distinguer les sujets sains des personnes à risque (maladie de Parkinson, déséquilibre postural). Cette étude valide la faisabilité d'un diagnostic embarqué sur smartphone dans des conditions d'usage domestique réelles, en dehors du laboratoire clinique. Le parallèle avec Securia est direct : dans les deux cas, un événement physique anormal (chute imminente ou arrachage) se traduit par une rupture caractéristique dans les signaux inertiels continus, détectable en temps réel sur un terminal mobile à ressources contraintes.

Ces travaux soulignent que le transfert méthodologique depuis la détection de chutes vers la détection de vols est non seulement plausible, mais constitue une piste concrète pour pallier le manque de données d'entraînement spécifiques aux arrachages réels.

2.11 JEUX DE DONNÉES ET LIMITES EXPÉRIMENTALES

2.11.1 PANORAMA DES JEUX DE DONNÉES DISPONIBLES

La disponibilité de jeux de données de qualité constitue un facteur limitant majeur dans ce domaine. Le tableau 2.5 présente un panorama des principaux jeux de données publics disponibles pour la HAR.

TABLEAU 2.5 : Panorama des jeux de données publics pour la HAR et la détection d'anomalies

Jeu de données	Capteurs	Activités	Sujets	Scénario vol
UCI HAR (Bayat *et al.*, 2014)	Acc, Gyro	6 normales	30	Non
WISDM (Kwapisz *et al.*, 2011)	Acc	6 normales	36	Non
MHEALTH	Acc, Gyro, Mag	12 activités	10	Non
SisFall	Acc, Gyro	Chutes + activités	38	Partiel
UniMiB-SHAR	Acc	17 activités + chutes	30	Non
Dataset Rivas-Caicedo *et al.* (2025)	Acc, Gyro, Orient.	12 activités	20	Non
Propriétaire Khan *et al.* (2010)	Acc, Gyro	Vols simulés	15	Oui (simulé)

2.11.2 LACUNES DES JEUX DE DONNÉES EXISTANTS ET DÉFIS DE COLLECTE

Les jeux de données proposés dans Bayat *et al.* (2014); Mekruksavanich & Jitpattanakul (2021); Kwapisz *et al.* (2011) et Rivas-Caicedo *et al.* (2025) sont majoritairement orientés vers

la reconnaissance d'activités normales et ne contiennent aucun scénario de vol. L'absence de dataset public dédié aux scénarios réalistes de vol constitue une lacune majeure. Elle oblige chaque équipe de recherche à construire son propre jeu de données expérimental, nuisant à la reproductibilité et à la comparabilité des approches proposées.

La collecte de données réalistes soulève également des difficultés méthodologiques et éthiques spécifiques. Simuler des vols en laboratoire introduit un biais de sélection. Recruter des participants pour des scénarios réels soulève des questions éthiques et logistiques. Ces défis justifient l'exploration de techniques d'augmentation de données et de génération synthétique par modèles génératifs (GANs, modèles de diffusion) comme pistes complémentaires pour enrichir les jeux d'entraînement.

2.12 SYNTHÈSE CRITIQUE GLOBALE ET LACUNES SCIENTIFIQUES

2.12.1 TABLEAU DE SYNTHÈSE COMPARATIVE GLOBALE

Le tableau présente une synthèse comparative de l'ensemble des travaux étudiés dans ce chapitre. Six critères d'évaluation sont retenus : la précision de détection, le taux de faux positifs (FP), la latence de réaction, la consommation énergétique, la robustesse aux variations comportementales et la portabilité entre modèles d'appareils.

TABLEAU 2.6 : Synthèse comparative globale des approches étudiées

Approche / Réf.	Précision	FP	Latence	Énergie	Robustesse	Portabilité
Réactive SIM Kanimozhi & Padmapriya (2021)	Basse	Très bas	Élevée	Faible	Basse	Haute
Détect. anomalies Thing *et al.* (2011)	Moyenne	Faible	Moyenne	Modérée	Moyenne	Haute

HAR classique Kwapisz *et al.* (2011)	Moyenne	Moyen	Moyenne	Modérée	Moyenne	Haute
HAR hiérarchique Khan *et al.* (2010)	Haute	Faible	Faible	Modérée	Moyenne	Moyenne
HAR wearable Politi *et al.* (2014a,b)	Haute	Faible	Faible	Modérée	Haute	Moyenne
Acc + Gyro Jalal *et al.* (2020); Yin *et al.* (2015)	Haute	Faible	Faible	Modérée	Haute	Moyenne
CBA personnalisé Hu (2015)	Haute	Moyen	Faible	Élevée	Haute	Basse
Multi-capteurs Rivas-Caicedo *et al.* (2025)	Haute	Faible	Faible	Modérée	Haute	Moyenne

Approche / Réf.	Précision	FP	Latence	Énergie	Robustesse	Portabilité
CNN Mekruksavanich & Jitpattanakul (2021)	Très haute	Faible	Faible	Élevée	Haute	Moyenne
LSTM Ordóñez & Roggen (2016); Chen & Xue (2015)	Très haute	Faible	Faible	Très élevée	Haute	Basse
Empreinte gyro. Tian *et al.* (2021)	Haute	Très bas	Moyenne	Faible	Très haute	Basse

RoadSense Allouch *et al.* (2017)	–	Faible	Faible	Faible	Haute	Haute
ML multicapteurs Nadeem *et al.* (2021)	Très haute	Faible	Faible	Élevée	Haute	Moyenne
Modèle proposé	Très haute	Très bas	Très faible	Modérée	Très haute	Haute

2.12.2 CONSTATS ISSUS DE L'ANALYSE CROISÉE

L'analyse croisée des références met en évidence plusieurs constats fondamentaux. Les solutions réactives (Kanimozhi & Padmapriya (2021); Thing *et al.* (2011)) restent fondamentalement limitées par leur paradigme post-hoc et ne peuvent pas constituer la base d'un système de prévention efficace. Les approches supervisées par apprentissage automatique classique (Kwapisz *et al.* (2011); Khan *et al.* (2010)) démontrent un fort potentiel en conditions contrôlées mais souffrent d'un manque de validité externe documenté. Les méthodes profondes (CNN, LSTM) offrent une modélisation temporelle nettement supérieure, mais imposent des contraintes énergétiques et computationnelles qui freinent leur déploiement embarqué en temps réel. L'authentification comportementale continue (Hu (2015)) ouvre une perspective prometteuse de personnalisation, mais reste confrontée à des défis non résolus en termes de dérive comportementale et d'optimisation énergétique.

2.12.3 IDENTIFICATION DE LA LACUNE SCIENTIFIQUE

L'analyse approfondie de la littérature révèle une lacune scientifique structurée autour de quatre axes complémentaires. Premièrement, aucun système existant ne

combine de manière intégrée la personnalisation comportementale individuelle et la détection multi-capteurs en temps réel. Deuxièmement, la question de l'optimisation énergétique pour le déploiement continu reste ouverte : les approches performantes sont trop coûteuses, et les approches économes sont insuffisamment précises. Troisièmement, la robustesse aux variations naturelles du comportement utilisateur n'est pas prise en compte de manière satisfaisante par les architectures existantes. Quatrièmement, l'absence de jeux de données publics réalistes constitue un frein majeur à la progression et à la comparabilité des travaux.

2.12.4 POSITIONNEMENT DU PRÉSENT MÉMOIRE ET ARCHITECTURE EN CASCADE PROPOSÉE

La figure 2.4 illustre le positionnement schématique du modèle proposé dans l'espace bidimensionnel précision-efficacité énergétique, par rapport aux approches existantes.

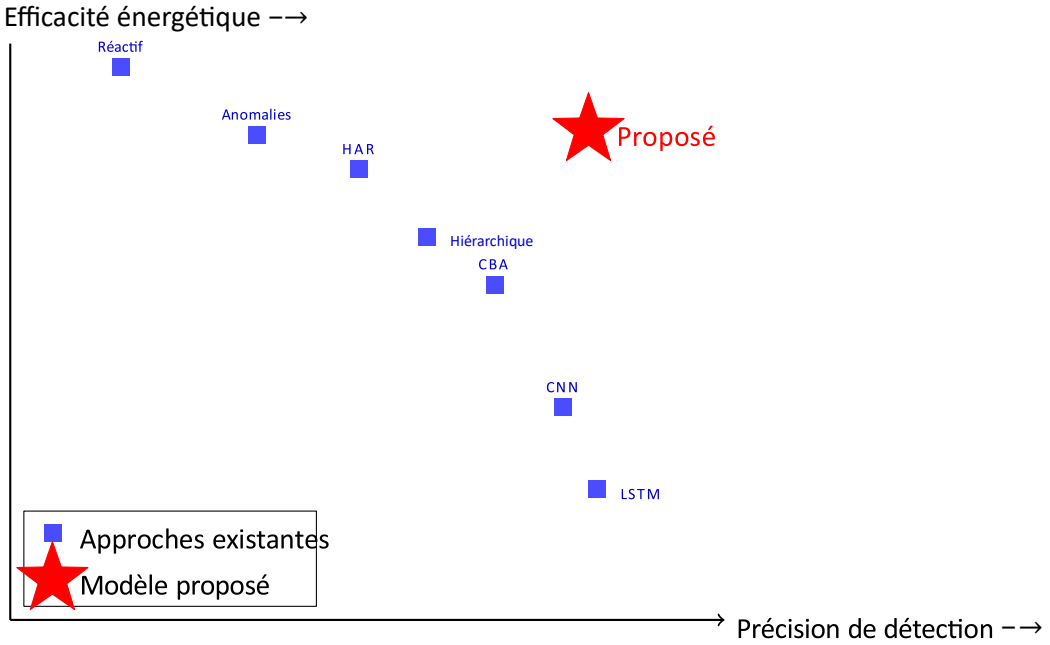


FIGURE 2.4 : Positionnement schématique du modèle proposé dans l'espace précision – efficacité énergétique.

Cette lacune scientifique justifie le développement d'un modèle hybride intelligent dédié spécifiquement à la détection proactive du vol de smartphone. Le modèle proposé vise à combiner les avantages des approches supervisées légères (faible latence, faible consommation) et des architectures profondes (haute précision, modélisation temporelle) au moyen d'une architecture en deux étages : un premier étage de détection rapide basé sur des règles heuristiques et un modèle léger, et un second étage de confirmation basé sur un réseau plus profond activé uniquement lors d'une alerte du premier étage. Cette architecture en cascade vise à concilier réactivité, précision et efficacité énergétique, répondant ainsi aux quatre axes de la lacune scientifique identifiée.

2.13 CONCLUSION

Ce chapitre a présenté une revue critique et structurée de la littérature sur la détection automatique de vols de smartphones par intelligence artificielle. Partant des approches réactives les plus élémentaires (Kanimozhi & Padmapriya (2021); Thing *et al.* (2011)), la revue a progressivement couvert les fondements du mobile sensing (Lane *et al.* (2010); Krichen (2021)), les techniques de reconnaissance d'activités humaines (Kwapisz *et al.* (2011); Khan *et al.* (2010); Politi *et al.* (2014a); Jalal *et al.* (2020); Yin *et al.* (2015); Politi *et al.* (2014b)), les méthodes de détection d'anomalies et d'authentification comportementale (Hu (2015); Krichen (2021); Chandola *et al.* (2009)), les architectures profondes (Mekruksavanich & Jitpattanakul (2021); Ordóñez & Roggen (2016); Chen & Xue (2015); Nadeem *et al.*

(2021)), les approches matérielles (Tian *et al.* (2021)) et les applications contextuelles connexes (Allouch *et al.* (2017); Rivas-Caicedo *et al.* (2025)).

L'analyse croisée a mis en évidence une lacune scientifique structurée autour de quatre axes : personnalisation comportementale, optimisation énergétique, robustesse aux variations naturelles et disponibilité de données réalistes. Cette lacune justifie et oriente le développement du système proposé dans les chapitres suivants.

Le chapitre III présentera la méthodologie de collecte de données, les choix d'ingénierie des caractéristiques et l'architecture du modèle hybride proposé, en réponse directe aux lacunes identifiées dans cette revue.

CHAPITRE III

MÉTHODOLOGIE ET CONCEPTION DU SYSTÈME

3.1 INTRODUCTION

Le chapitre précédent a mis en évidence une lacune scientifique structurée autour de quatre axes : l'absence de modélisation comportementale entraînée sur des données de vol réelles, le compromis non résolu entre précision et efficacité énergétique, la robustesse insuffisante aux variations naturelles du comportement et le manque de jeux de données réalistes. Le présent chapitre décrit en deux parties complémentaires la réponse apportée à ces lacunes.

La première partie expose la méthodologie d'apprentissage automatique : collecte et préparation des données, ingénierie des

caractéristiques et conception du pipeline de détection hybride. La seconde partie présente la conception complète du système Android : architecture applicative, modules fonctionnels, intégration Firebase et contraintes techniques de déploiement.

La figure 3.1 illustre le pipeline méthodologique global, depuis la collecte des données brutes jusqu'au déploiement embarqué.

FIGURE 3.1 : Pipeline méthodologique global du système Securia.

3.2 MÉTHODOLOGIE D'APPRENTISSAGE AUTOMATIQUE

3.2.1 PRINCIPES FONDAMENTAUX

Trois principes fondamentaux guident l'ensemble des choix méthodologiques.

Le premier est la détection en cascade : un Isolation Forest léger assure une surveillance continue à faible coût computationnel, et un Autoencoder Dense est activé uniquement en cas de suspicion détectée par le premier étage. Cette approche réconcilie réactivité et efficacité énergétique, résolvant ainsi le compromis identifié comme lacune centrale dans la littérature.

Le second est l'apprentissage sur données de vol : le modèle est pré-entraîné sur un corpus incluant des activités normales et des scénarios de vol simulés, sans calibration individuelle par l'utilisateur. Le modèle pré-entraîné est déployé directement sur l'appareil.

Le troisième est le traitement entièrement local : conformément aux principes du *Privacy by Design,*

l'ensemble du traitement des données sensibles est effectué exclusivement sur l'appareil, sans transfert vers des serveurs distants pendant la phase de surveillance continue.

3.2.2 SOURCES DE DONNÉES

La stratégie de collecte repose sur deux sources complémentaires couvrant à la fois les comportements normaux et les scénarios de vol.

JEU DE DONNÉES MOTIONSENSE (SUB_1 À SUB_5)

Le jeu de données public MotionSense fournit des enregistrements collectés lors d'activités quotidiennes standardisées. Les fichiers sub_1.csv à sub_5.csv contiennent les colonnes suivantes, selon les conventions de l'API CoreMotion d'Apple :

— attitude.roll, attitude.pitch, attitude.yaw — orientation de l'appareil (rad)

— gravity.x/y/z — composantes gravitationnelles (g)

— rotationRate.x/y/z — vitesse angulaire (rad/s)

— userAcceleration.x/y/z — accélération utilisateur sans gravité (g)

JEUX DE DONNÉES PROPRIÉTAIRES SECURIA_DATASET

Deux fichiers propriétaires ont été constitués spécifiquement pour le projet Securia, ciblant chacun un mode d'activité distinct :

— SecurIA_Dataset_20260225_124122.csv — mode marche, contenant les colonnes Accelerometer_x/y/z, Gyroscope_x/y/z, latitude et longitude

— SecurIA_Dataset_running.csv — mode course, même structure, signaux d'intensité plus élevée

Ces fichiers incluent des enregistrements d'activités normales ainsi que des scénarios simulés d'arrachage selon plusieurs modalités (frontal, latéral, depuis une table).

3.2.3 PROCÉDURE D'ÉTIQUETAGE ET CONSTITUTION DES ENSEMBLES DE DONNÉES

PROTOCOLE D'ÉTIQUETAGE DU SECURIA_DATASET

Chaque observation du SecurIA_Dataset est associée à deux colonnes d'annotation ajoutées manuellement lors de la collecte :

— label : valeur binaire indiquant la nature de l'observation. 0 désigne une activité normale (marche ou course régulière, mise en poche rapide, activité sportive intense, conduite sur route cahoteuse, chute accidentelle). 1 désigne un scénario de vol simulé.

— category : catégorie fine de l'observation, prenant l'une des valeurs suivantes : normal, arrachage_frontal, arrachage_lateral, arrachage_table. Cette granularité permet une analyse par modalité de vol lors de l'évaluation.

L'annotation a été réalisée par l'auteur de manière simultanée à la collecte : chaque session d'enregistrement était dédiée à une seule catégorie (par exemple, une session de marche normale, puis une session

d'arrachages frontaux), ce qui permet d'affecter le label correspondant à toutes les observations de la session sans ambiguïté. Aucun algorithme d'annotation automatique n'a été utilisé; les étiquettes sont donc de source humaine et vérifiées visuellement par inspection des profils de signal IMU caractéristiques de chaque modalité.

CONSTITUTION DES ENSEMBLES D'ENTRAÎNEMENT ET DE TEST

La séparation entraînement/évaluation repose sur la nature du modèle utilisé, qui est non supervisé au sens de l'entraînement (voir § 4.3).

Ensemble d'entraînement uniquement des observations normales (label = 0). L'Isolation Forest et l'Autoencoder apprennent exclusivement la distribution statistique des comportements normaux, sans jamais être exposés aux scénarios de vol. La répartition effective est la suivante :

TABLEAU 3.1 : Constitution des ensembles d'entraînement et de test par mode

Mode	Total	Entraîn. (label=0)	Test (label=1)	Taux d'anomalie
Marche	38869	36925	1944	5,0%
Course	29432	27960	1472	5,0%
Total	68301	64885	3416	5,0%

Ensemble d'évaluation l'ensemble complet (observations normales + scénarios de vol, soit 68301 observations) est soumis aux modèles entraînés pour calculer les métriques de performance. Les étiquettes (label) servent uniquement à calculer VP, VN, FP et FN; les modèles n'y ont jamais accès pendant l'inférence.

Ce protocole correspond à la pratique standard en détection d'anomalies : le modèle apprend le comportement normal sans connaître les anomalies, et la performance est mesurée *a posteriori* sur un ensemble annoté incluant les deux classes (Chandola *et al.*, 2009).

3.2.4 FUSION ET PRÉPARATION DES DONNÉES

Les deux sources de données présentent des structures de colonnes différentes. Elles sont combinées par concaténation verticale après sélection des colonnes pertinentes dans chaque source. Les valeurs manquantes générées par cette concaténation hétérogène sont imputées par la médiane de chaque colonne, stratégie robuste aux valeurs atypiques pour des données de capteurs continus.

TABLEAU 3.2 : Structure du DataFrame consolidé pour le mode course (29432 observations)

Source	Colonnes retenues	Traitement
SecurIA_Dataset_running	Acc_x/y/z, Gyro_x/y/z, lat, long (8 col.)	Sélection directe
sub_1 à sub_5	attitude, gravity, rotationRate, userAcceleration (12 col.)	Suppression Unnamed :0
Total		20 colonnes, imputation médiane

3.2.5 NORMALISATION

Une normalisation par standardisation z-score est appliquée via le StandardScaler de scikit-learn sur l'ensemble des colonnes :

$$x_i - \mu$$

$$\hat{x}_i = \frac{}{\sigma} \tag{3.1}$$

Le scaler est ajusté (*fit*) exclusivement sur les données d'entraînement et appliqué (*transform*) sans réajustement sur les données de production, garantissant l'absence de fuite d'information entre les phases.

3.2.6 CALCUL DES CARACTÉRISTIQUES SMA

La SMA (*Signal Magnitude Area*) constitue la caractéristique centrale des deux modèles. Elle est invariante à l'orientation du capteur, propriété essentielle pour un système déployé sur des smartphones tenus de manières diverses.

Deux variantes ont été implémentées selon le mode d'activité.
Mode marche — magnitude instantanée calculée sur chaque échantillon :

$$\text{SMA\_acc} = \sqrt{a^2_x + a^2_y + a^2_z}, \qquad \text{SMA\_gyr} = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{3.2}$$

Mode course — moyenne glissante sur une fenêtre de 10 échantillons appliquée sur la magnitude calculée, pour lisser les oscillations périodiques plus marquées :

$$\text{SMA\_acc}^{(t)} = \frac{1}{10} \sum_{i=t-9}^{t} \sqrt{a_x^{(i)2} + a_y^{(i)2} + a_z^{(i)2}} \tag{3.3}$$

3.2.7 JUSTIFICATION DES CHOIX TECHNOLOGIQUES

Avant de détailler l'architecture de détection, il est essentiel de justifier les choix technologiques qui en constituent la base. Chaque composante du pipeline de détection résulte d'une décision délibérée parmi plusieurs alternatives viables. Cette sous-section documente les critères de décision et les raisons pour lesquelles les technologies retenues ont été préférées à leurs concurrentes, afin d'assurer la reproductibilité et la traçabilité des décisions de conception.

ALGORITHME DE DÉTECTION : CASCADE IF + AUTOENCODER DENSE

TABLEAU 3.3 : Comparaison des architectures de détection d'anomalies pour déploiement embarqué

Critère	IF + AE Dense	LSTM	SVM	Règles heuristiques seules
Taille du modèle	< 3 Ko	> 1 Mo	Variable	0 Ko
Latence inférence ARM	< 0,5 ms	50–200 ms	< 5 ms	< 0,1 ms
Apprentissage non supervisé	Oui	Non	Non	Non
Robustesse aux variations	Haute (P95)	Haute	Moyenne	Faible
Convertible en TFLite	Oui (AE)	Oui	Non natif	Non applicable
Données étiquetées requises	Non	Oui	Oui	Non
Verdict	Retenu	Non retenu	Non retenu	Non retenu

L'architecture LSTM aurait offert une meilleure modélisation des dépendances temporelles dans les séquences de capteurs, mais son coût computationnel (50 à 200 ms sur ARM, plusieurs Mo de modèle) est incompatible avec une surveillance continue sur batterie. Le SVM supervisé exige des données de vol étiquetées difficiles à collecter. L'approche par règles heuristiques seules manque de robustesse aux variations inter-individuelles. La cascade Isolation

Forest + Autoencoder Dense résout ces quatre contraintes : non supervisé, ultra-compact (<

3 Ko), latence mesurée à 0,014 ms, et robustesse assurée par le seuil au 95^{e} percentile.
FORMAT DE MODÈLE EMBARQUÉ : TENSORFLOW LITE

TABLEAU 3.4 : Comparaison des formats de déploiement de modèles ML embarqués

Critère	TFLite	ONNX Run-time	PyTorch Mobile
Taille du runtime	≈ 1 Mo	≈ 6 Mo	≈ 20 Mo
Intégration An-droid	Natif + AAR	Manuelle	Manuelle
Support Keras/TF	Direct	Via conversion	Via conversion
Délégué NNA-PI/GPU	Oui	Partiel	Limité
Latence sur ARM	< 1 ms (dense)	2–5 ms	5–15 ms
Maturité	Haute	Moyenne	Haute
Verdict	Retenu	Non retenu	Non retenu

TensorFlow Lite s'impose comme le choix naturel compte tenu du pipeline d'entraînement TensorFlow/Keras utilisé dans les notebooks. La conversion directe via TFLiteConverter.from_keras_model() préserve la précision numérique et

produit des fichiers de 1,5 à 2,8 Ko, plusieurs ordres de grandeur sous les runtimes ONNX ou PyTorch Mobile. Le délégué XNNPACK activé par défaut assure une optimisation automatique sur les processeurs ARM Cortex-A
courants.

3.2.8 ARCHITECTURE DU MODÈLE DE DÉTECTION HYBRIDE

VUE D'ENSEMBLE

Le modèle de détection combine deux algorithmes complémentaires : un Isolation
Forest pour la détection rapide de premier étage et un Autoencoder Dense pour la confirmation
de second étage. L'architecture Dense est retenue plutôt qu'une architecture
récurrente LSTM pour deux raisons pratiques déterminantes : la latence
d'inférence (2–5 ms vs 50–200 ms sur ARM) et la taille du modèle (≈ 3 Ko vs
plusieurs Mo après conversion TFLite), conditions indispensables au
déploiement embarqué continu.

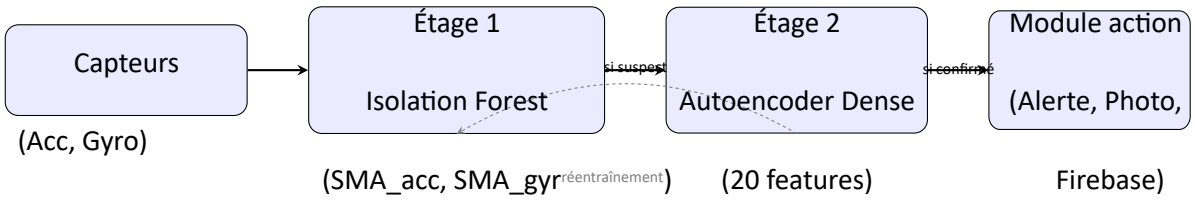


FIGURE 3.2 : Architecture en cascade du système Securia.

ÉTAGE 1 ISOLATION FOREST

L'Isolation Forest (Chandola *et al.*, 2009) détecte les anomalies par isolation aléatoire.

Le score d'anomalie est défini par :

$$s(x,n) = 2^{-\frac{E[h(x)]}{c(n)}} \quad (3.4)$$

Un score proche de 1 indique une anomalie; proche de 0,5 une observation normale.

L'étiquette −1 est assignée aux anomalies, +1 aux observations normales.

Pour le mode marche, un système de catégorisation en quatre niveaux est appliqué sur les anomalies détectées en fonction de seuils empiriques définis sur les valeurs de SMA :

TABLEAU 3.5 : Seuils de catégorisation des anomalies mode marche

Catégorie	SMA_acc	SMA_gyr	Priorité
Critique	< 0,5 ou > 20	< 0,01 ou > 7,5	Maximum
Anomalie	> 15	> 5	Haute
Alerte	> 10	> 1	Modérée
Normal	Autres		Aucune

La figure 3.3 illustre la distribution des anomalies détectées par l'Isolation Forest dans l'espace bidimensionnel SMA_acc / SMA_gyr sur les données du mode marche. Les activités normales (vert) se concentrent autour de l'origine avec des valeurs faibles de magnitude, confirmant que l'utilisateur en marche produit des signaux stables et réguliers. Les alertes

modérées (orange) se dispersent vers des valeurs intermédiaires, caractéristiques de mouvements plus brusques mais non suspects. Les anomalies confirmées (rouge) et les cas critiques (violet) présentent des valeurs élevées sur l'un ou l'autre axe voire les deux — traduisant des accélérations ou rotations inhabituelles caractéristiques d'un arrachage. Cette séparation visuelle valide la pertinence discriminante des caractéristiques SMA pour la détection de vols.

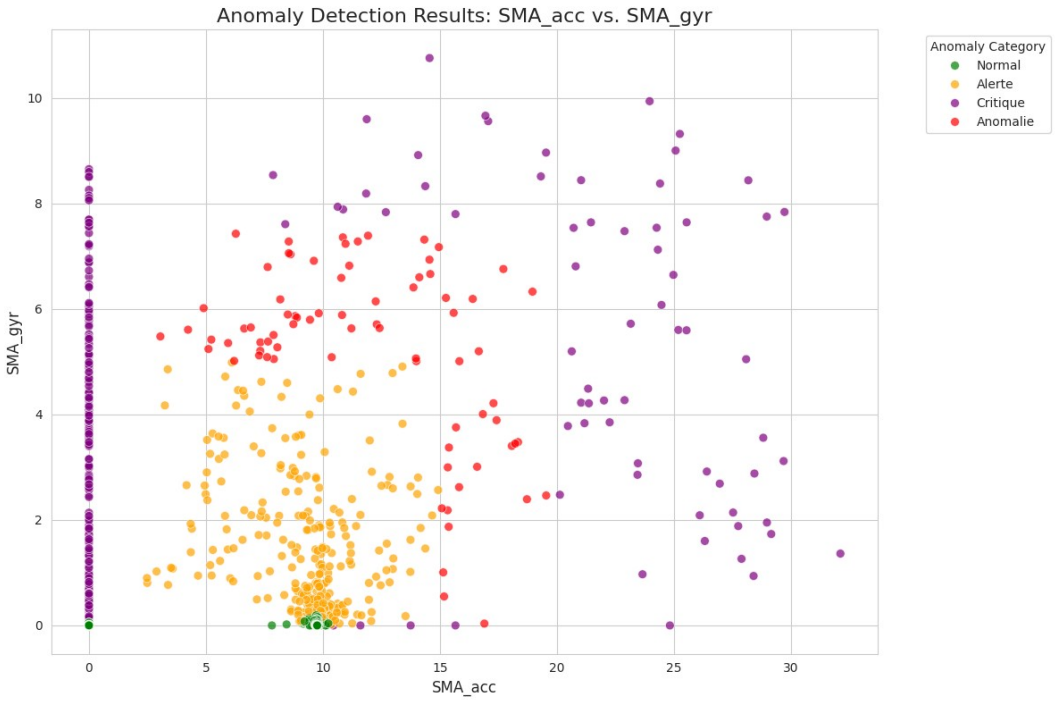


FIGURE 3.3 : Résultats de la détection d'anomalies par Isolation Forest dans l'espace SMA_acc / SMA_gyr (mode marche). Les points verts correspondent aux activités normales, les points orange aux alertes modérées, les points rouges aux anomalies confirmées et les points violets aux anomalies critiques.

ÉTAGE 2 AUTOENCODER DENSE

L'Autoencoder apprend à reconstruire les données normales; les observations dont l'erreur de reconstruction MSE dépasse le seuil ϑ sont classifiées comme anomalies. Deux architectures sont déployées :

TABLEAU 3.6 : Architectures des Autoencoders Dense par mode d'activité

Couche	Mode marche	Mode course	Paramètres
Entrée	(2,)	(20,)	
Encodeur	1 unité	10 unités	ReLU
Décodeur	2 unités	20 unités	Linéaire
Optimiseur		Adam, lr=10^{-3}	50 époques, batch 32/128

Le seuil de détection est fixé au 95^{e} percentile de la distribution des erreurs de reconstruction sur les données d'entraînement :

$$\vartheta = P_{95}(\text{MSE}(\hat{x},x))= 0{,}3926 \qquad (3.5)$$

LOGIQUE DE DÉCISION EN CASCADE

TABLEAU 3.7 : Logique de décision du système de détection en cascade

Label IF	Décision étage 1	Action étage 2	Résultat
+1	Normal	Non activé	Aucune alerte
−1	Suspect	Autoencoder actif	MSE calculée
−1, MSE < ϑ	Non confirmé		Aucune alerte

−1, MSE ≥ ϑ	Vol confirmé		Alerte

3.2.9 CONVERSION TENSORFLOW LITE

Les deux Autoencoders sont convertis via TFLiteConverter.from_keras_model() et sauvegardés dans deux fichiers :

— autoencoder_anomaly_model.tflite — mode marche, Dense(2→1→2)

— detection_course.tflite — mode course, Dense(20→10→20)

La taille de chaque fichier est de l'ordre de 3 Ko, garantissant un chargement quasi instantané au démarrage du service Android.

3.3 MODÉLISATION UML DU SYSTÈME SECURIA

La présente section formalise la conception du système Securia à travers trois modèles UML complémentaires : le diagramme de cas d'utilisation, qui décrit les interactions entre les acteurs et le système; le diagramme de classes simplifié, qui représente la structure statique de l'application Android; et le diagramme de séquence, qui illustre le déroulement dynamique d'un cycle de détection de vol.

3.3.1 DIAGRAMME DE CAS D'UTILISATION

Le diagramme de cas d'utilisation (figure 3.4) présente les principales fonctionnalités de Securia du point de vue de ses deux acteurs : l'Utilisateur légitime (propriétaire du smartphone) et le Système (traitements automatiques déclenchés sans intervention humaine).

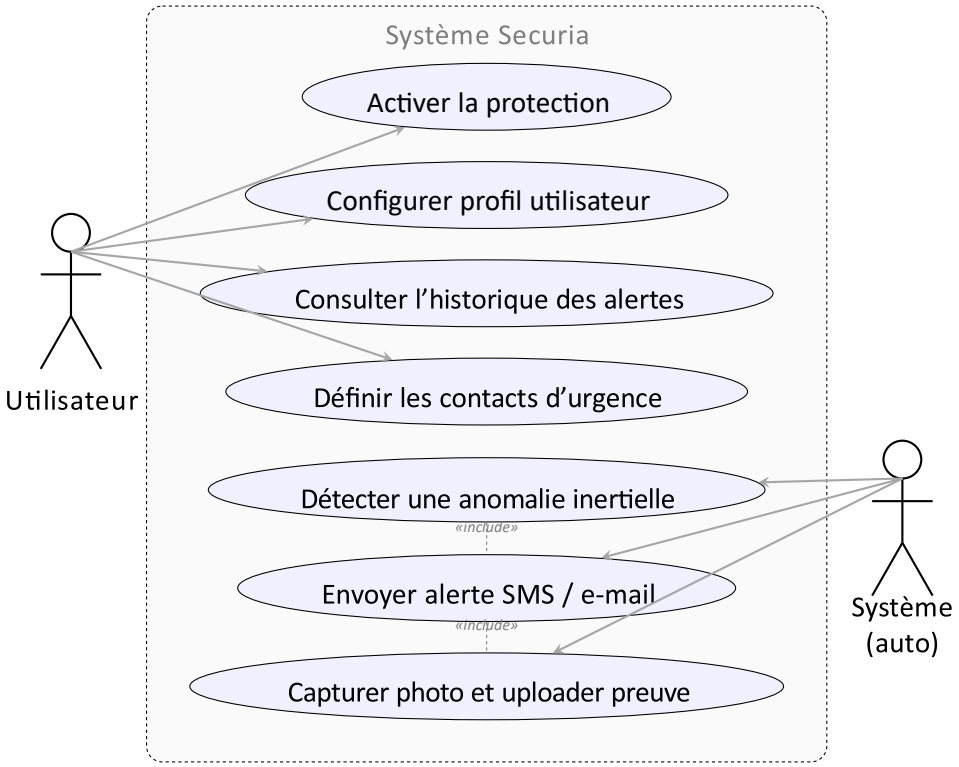


FIGURE 3.4 : Diagramme de cas d'utilisation du système Securia.

3.3.2 DIAGRAMME DE CLASSES SIMPLIFIÉ

Le diagramme de classes (figure 3.5) représente les principales classes de l'application Android et leurs relations. Les dépendances sont représentées par des flèches en pointillé, les associations par des flèches pleines.

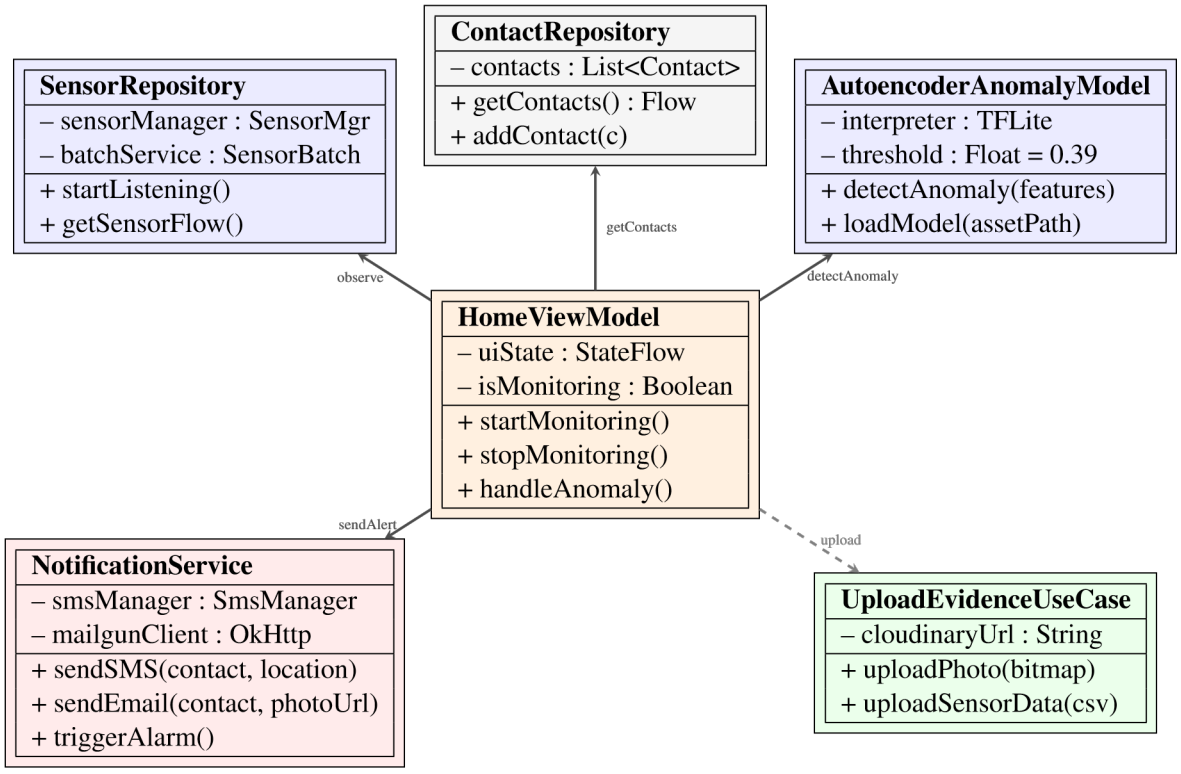


FIGURE 3.5 : Diagramme de classes simplifié de l'application Securia (Android).

3.3.3 DIAGRAMME DE SÉQUENCE CYCLE DE DÉTECTION D'UN VOL

Le diagramme de séquence (figure 3.6) décrit le déroulement complet d'un cycle de détection, depuis la réception des données des capteurs jusqu'au déclenchement des contremesures. Chaque échange de messages correspond à un appel de méthode réel dans le code Kotlin.

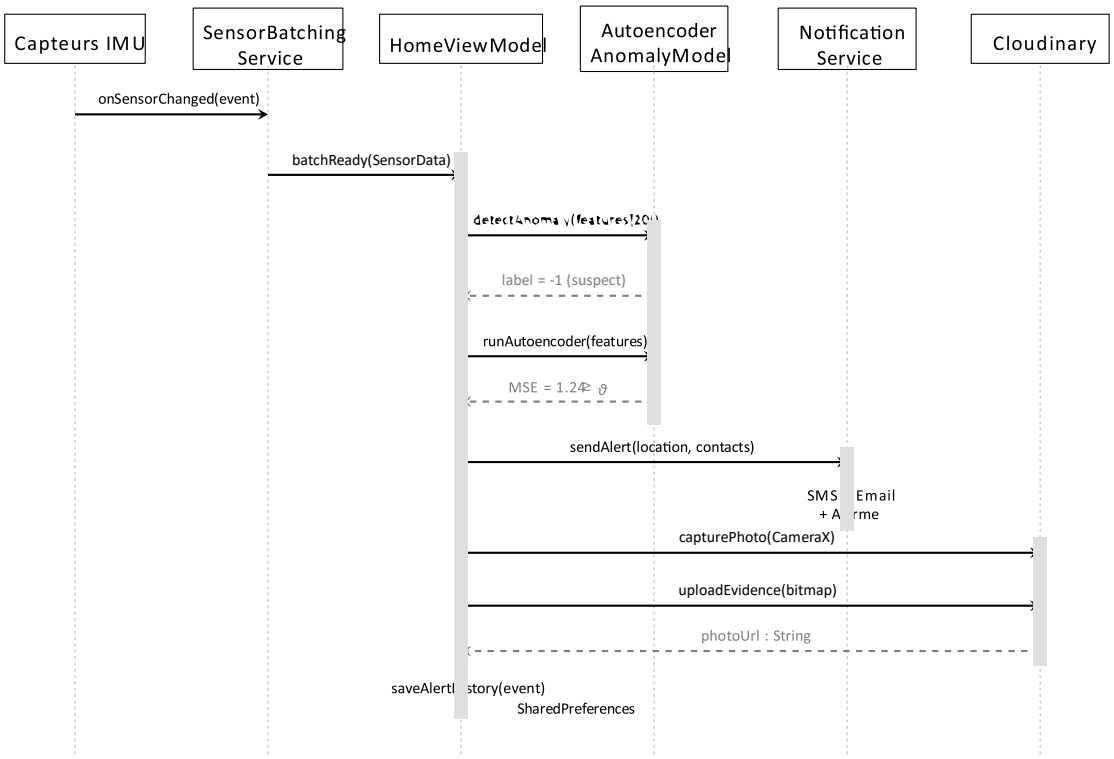


FIGURE 3.6 : Diagramme de séquence cycle complet de détection d'un vol dans Securia.

3.4 CONCEPTION DU SYSTÈME ANDROID

3.4.1 ARCHITECTURE DE L'APPLICATION

L'application Securia est développée en Kotlin avec Jetpack Compose et suit le patron d'architecture MVVM (*Model-View-ViewModel*), organisé en modules par fonctionnalité (*feature-based*).

TABLEAU 3.8 : Couches de l'architecture MVVM de Securia

Couche	Rôle	Exemples clés
View	Écrans Jetpack Compose, navigation	HomeScreen, LocationScreen, LoginScreen

ViewModel	Logique métier, état via StateFlow	HomeViewModel, LoginViewModel, SplashViewModel
Model/Data	Accès données, services, repositories	SensorRepository, AutoencoderAnomalyModel
Domain	Use Cases, interfaces repositories	StoreSensorDataUseCase, UploadToFirebaseUseCase

La structure des packages est organisée ainsi :

```
com.mobile.securia/
+-- core/                    # Transversal : utils, DI, domain, UI partagée
+-- feature/
| +-- detection/             # AutoencoderAnomalyModel, HomeViewModel, HomeScreen
| +-- sensors/               # SensorBatchingService, SensorRepository, SensorData
| +-- alerts/                # NotificationService, SimpleNotificationService
| +-- analytics/             # StoreSensorDataUseCase, UploadToFirebaseUseCase
| +-- auth/                  # Login, Register (ViewModel, State, Screen)
| +-- contacts/              # ContactRepository, Contact model
| +-- location/              # LocationViewModel, Geofencing, zones GPS
| +-- voice/                 # VoiceRecognitionService (ForegroundService)
| \-- settings/              # SettingsScreen
+-- di/                      # AppDependencies (injection manuelle)
\-- ui/theme/                # Material 3
```

L'injection de dépendances est gérée manuellement via l'objet singleton AppDependencies, avec une migration future prévue vers Hilt ou Koin.

3.4.2 COMMUNICATION INTER-MODULES

Le HomeViewModel joue le rôle d'orchestrateur central. Il coordonne l'ensemble des modules via trois mécanismes :

— StateFlow/Flow : communication réactive entre ViewModel et UI, et entre Repository et ViewModel

— Appels directs : le HomeViewModel invoque directement les services (SensorRepository, NotificationService, etc.)

— viewModelScope.launch : toutes les opérations asynchrones utilisent les coroutines Kotlin dans le scope du ViewModel

La figure 3.7 illustre les flux d'interactions entre les modules.

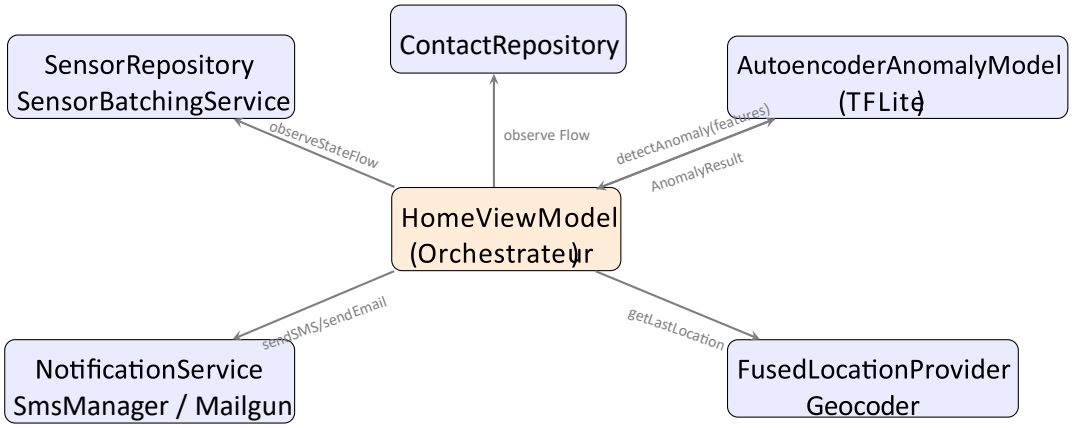


FIGURE 3.7 : Communication inter-modules orchestrée par le HomeViewModel.

3.4.3 MODULE DE COLLECTE DES CAPTEURS

La collecte est assurée par le SensorBatchingService, qui enregistre simultanément les événements de l'accéléromètre (TYPE_ACCELEROMETER) et du gyroscope (TYPE_GYROSCOPE) via le SensorManager Android.

Les événements sont accumulés dans deux files d'attente ConcurrentLinkedQueue thread-safe. Le flush vers le fichier CSV local s'effectue tous les 200 échantillons pour minimiser les opérations d'écriture disque. Le format CSV produit est le suivant :

session_id, epoch, Acc_x, Acc_y, Acc_z, Gyro_x, Gyro_y, Gyro_z, latitude, longitude, label, category

TABLEAU 3.9 : Paramètres de configuration du SensorBatchingService

Paramètre	Valeur
Taille lot UI	10 échantillons (mise à jour temps réel)
Seuil flush CSV	200 échantillons (écriture disque)
Latence max rapport	1 seconde (1000000 µs)
Fréquence	SENSOR_DELAY_NORMAL
Support FIFO	Vérifié via sensor.fifoMaxEventCount

3.4.4 MODULE DE DÉTECTION IA SUR ANDROID

Le module de détection est implémenté dans la classe AutoencoderAnomalyModel.kt. Il opère sur une fenêtre glissante de 50 échantillons et extrait un vecteur de 20 caractéristiques :

— Moyenne et écart-type de l'accéléromètre (axes X, Y, Z) 6 valeurs
— Moyenne et écart-type du gyroscope (axes X, Y, Z) 6 valeurs

— Moyenne et écart-type de la magnitude de l'accéléromètre 2 valeurs

— Moyenne et écart-type de la magnitude du gyroscope 2 valeurs

— Maximum et minimum des magnitudes accéléromètre et gyroscope 4 valeurs

Ce vecteur est normalisé par le StandardScaler (moyennes et écarts-types chargés depuis les assets), puis soumis à l'interpréteur TFLite. La MSE de reconstruction est comparée au seuil $\vartheta = 0{,}3926$ pour la décision finale. L'ensemble du pipeline d'inférence s'exécute en moins de 5 ms sur ARM.

TABLEAU 3.10 : Paramètres du module de détection IA sur Android

Paramètre	Valeur
Taille d'entrée	20 features
Fenêtre glissante	50 échantillons
Seuil MSE	0,3926
Modèles disponibles	autoencoder_anomaly_model.tflite, detection_course.tflite
Normalisation	StandardScaler (means/stds configurables)

3.4.5 MODULE D'ACTION ET MODES DE PROTECTION

Securia propose six modes de protection activables indépendamment :

TABLEAU 3.11 : Modes de protection disponibles dans Securia

Mode	Variable d'état	Description
Marche	walkingModeEnable	dDétection IA via accéléromètre/gyroscope
Course	runningModeEnable	dDétection de vol pendant la course
Zone GPS	perimeterModeEnab	ledAlerte si sortie du périmètre géofencé
Écriture	writingModeEnable	dDétection pendant l'écriture
Utilisation	phoneUsageModeEna	bledProtection pendant usage actif
Poche/Main	pocketModeEnabled	Détection d'arrachement de poche

Lors d'une détection confirmée, la chaîne d'action suivante se déclenche :

1. Notification push haute priorité (CATEGORY_ALARM, canal IMPORTANCE_HIGH)
2. SMS automatique aux contacts d'urgence via SmsManager avec la localisation GPS (configurable via isAutoAlertsEnabled())
3. Email automatique via l'API Mailgun (OkHttp, Basic Auth) avec lien Google Maps vers la position GPS
4. Vibration pattern createWaveform(500-200-500 ms) (configurable via isVibrationAlertsEnabl
5. Son d'alarme RingtoneManager.TYPE_ALARM (configurable via isSoundAlertsEnabled())
6. Enregistrement dans l'historique des alertes (JSON, SharedPreferences)
7. Géocodage inverse pour inclure l'adresse lisible dans le message

3.4.6 INTÉGRATION FIREBASE

L'application intègre deux services Firebase et un service tiers pour la gestion des preuves visuelles, à des degrés différents d'avancement :

TABLEAU 3.12 : État d'intégration des services cloud dans Securia

Service	Utilisation	État
Firebase Auth	Authentification login/register, langue FR	Implémenté
Firebase Firestore	Stockage des lectures capteurs comme preuves numériques via collection('sensor_readings').add()	En cours (TODO)
Cloudinary	Stockage sécurisé des photos de preuve capturées lors d'un vol confirmé, via l'API REST Cloudinary (upload signé)	Intégré

SecurIAApplication initialise Firebase au démarrage via FirebaseApp.initializeApp() et configure la langue française. Le module d'upload

des photos de preuve a initialement été conçu avec Firebase Storage; cependant, les contraintes du palier gratuit de Firebase Storage (quota de bande passante limité et facturation par opération au-delà du seuil) l'ont rendu inadapté pour une application de sécurité susceptible de capturer de nombreuses images lors d'alertes fréquentes. Cloudinary a été retenu comme alternative : son palier gratuit offre 25 Go de stockage et 25 Go de bande passante mensuelle, dispose d'une API REST directement consommable depuis Android via OkHttp, et permet une compression automatique des images avant stockage, réduisant ainsi la bande passante consommée. L'UploadToFirebaseUseCase a été renommé UploadEvidenceUseCase et redirige désormais l'upload vers l'endpoint Cloudinary.

3.4.7 CYCLE DE VIE DU FOREGROUNDSERVICE

Le VoiceRecognitionService est le seul ForegroundService déclaré dans le manifest. Son cycle de vie comprend quatre états : *Arrêté* → *Démarré* → *Au premier plan* → *Écoute active*. Il est déclaré avec foregroundServiceType="microphone" (requis Android 14+) et requiert les permissions FOREGROUND_SERVICE_MICROPHONE et RECORD_AUDIO. Un WAKE_LOCK maintient le CPU actif pendant l'écoute pour éviter toute interruption de la surveillance vocale.

3.5 CONTRAINTES TECHNIQUES DE CONCEPTION

3.5.1 GESTION DE LA BATTERIE SENSOR BATCHING

L'optimisation énergétique repose sur l'API *Sensor Batching* d'Android, qui accumule les données dans le FIFO matériel du capteur et ne réveille le CPU qu'une fois le lot complet.

Cinq techniques complémentaires sont mises en œuvre :

TABLEAU 3.13 : Techniques d'optimisation énergétique implémentées

Technique	Implémentation
Sensor Batching API	maxReportLatency = 1s registerListener() via
FIFO matériel	Vérification via sensor.fifoMaxEventCount
Buffer CSV	Écriture disque tous les 200 échantillons uniquement
ConcurrentLinkedQueue	Files thread-safe sans blocage
BatteryOptimizationStats	Métriques temps réel, économie estimée 50–90%

3.5.2 SÉCURITÉ DES DONNÉES

TABLEAU 3.14 : Mesures de sécurité implémentées dans Securia

Aspect	Implémentation
Chiffrement préférences	EncryptedSharedPreferences (androidx.security:security-crypto)
Verrouillage PIN	Code PIN stocké via PreferencesHelper
Device Admin	SecuriaDeviceAdminReceiver pour verrouillage distant

Auto-lock	Durée configurable, défaut 60 s
Firebase Auth	Authentification sécurisée login/register
SMTP sécurisé	TLS (STARTTLS) pour l'envoi email

3.5.3 COMPATIBILITÉ ET PERMISSIONS ANDROID

L'application cible Android 7.0 (minSdk 24) à Android 14 (targetSdk 34). Le ta-

bleau 3.15 liste les permissions essentielles.

TABLEAU 3.15 : Permissions Android déclarées dans le manifest de Securia

Permission	Usage
ACCESS_FINE_LOCATION	Localisation GPS précise
ACCESS_BACKGROUND_LOCATION	Géofencing en arrière-plan
CAMERA	Capture de photo comme preuve
HIGH_SAMPLING_RATE_SENSORS	Capteurs haute fréquence
RECORD_AUDIO	Reconnaissance vocale
FOREGROUND_SERVICE_MICROPHONE	Service micro (Android 14+)
WAKE_LOCK	Maintien CPU actif
SEND_SMS	Envoi SMS automatique
BIND_DEVICE_ADMIN	Verrouillage d'écran

L'accéléromètre est déclaré comme matériel obligatoire

(android.hardware.sensor.acceleromete le gyroscope et la caméra comme optionnels. Les dépendances principales du projet sont :

TensorFlow Lite (runtime, support, metadata), Jetpack Compose + Material 3, Room (KSP),

Firebase (Auth, Firestore), Cloudinary (upload preuves), CameraX, Google Maps Compose, OkHttp, WorkManager et androidx.security:security-crypto.

LANGAGE DE DÉVELOPPEMENT : KOTLIN

TABLEAU 3.16 : Comparaison des langages de développement Android

Critère	Kotlin	Java	Flutter/Dart
Support officiel Google	Recommandé (1er)	Maintenu	Partiel
Coroutines natives	Oui (intégrées)	Non (Threads/Rx-Java)	Oui (async/await)
Null safety	Natif	Non	Natif
Jetpack Com-pose	Natif	Limité	Non applicable
Interopérabilité JVM	Complète	Complète	Non
Accès capteurs Android	Direct	Direct	Via plugins
Verdict	Retenu	Non retenu	Non retenu

Kotlin a été retenu pour trois raisons déterminantes : son support de première classe par Google depuis 2017, ses coroutines natives qui simplifient considérablement la gestion de la surveillance asynchrone des capteurs, et la compatibilité native avec Jetpack Compose pour l'interface déclarative. Java aurait requis l'adoption de bibliothèques tierces (RxJava, Executor) pour atteindre un niveau équivalent de réactivité.

ARCHITECTURE APPLICATIVE : MVVM

TABLEAU 3.17 : Comparaison des patrons d'architecture Android

Critère	MVVM	MVC	MVP
Séparation UI / logique	Forte	Faible	Moyenne
Support Jetpack Compose	Natif (State-Flow)	Non	Partiel
Testabilité du View-Model	Élevée	Faible	Moyenne
Gestion du cycle de vie	ViewModel survit à la rotation	Manuel	Manuel
Recommandation Google	Oui (Architecture Guide)	Non	Non
Verdict	Retenu	Non retenu	Non retenu

MVVM est le patron recommandé par Google pour les applications Android modernes. Il s'intègre naturellement avec Jetpack Compose via les flux réactifs StateFlow, et le cycle de vie du ViewModel qui survit aux rotations d'écran évite de relancer la surveillance des capteurs à chaque changement de configuration, un avantage critique pour une application de sécurité fonctionnant en continu.

STOCKAGE DES PREUVES VISUELLES : CLOUDINARY

TABLEAU 3.18 : Comparaison des solutions de stockage d'images pour les preuves

Critère	Cloudinary	Firebase Storage	AWS S3

Quota gratuit	25 Go / mois	5 Go (limité)	5 Go (12 mois)
SDK Android	API REST (OkHttp)	SDK natif	SDK officiel
Compression auto	Oui (intégrée)	Non	Non
Coût au-delà du quota	Faible	Élevé par opération	Faible
Intégration Firebase	Indépendante	Couplée	Indépendante
Privacy by Design	Compatible	Compatible	Compatible
Verdict	Retenu	Non retenu	Non retenu

Firebase Storage a été initialement envisagé pour sa cohérence avec Firebase Auth et Firestore. Cependant, son modèle de tarification facturation par opération au-delà d'un quota mensuel faible est inadapté à une application de sécurité susceptible de générer de nombreuses captures en cas d'activité criminelle intense. Cloudinary offre 25 Go de stockage et de bande passante mensuels gratuitement, une API REST simple consommable via OkHttp déjà présent dans les dépendances, et une compression automatique des images capturées avant upload.

3.6 CONCLUSION

Ce chapitre a présenté la méthodologie et la conception complètes du système Securia. Sur le plan algorithmique, l'architecture hybride Isolation Forest + Autoencoder Dense répond directement à la lacune identifiée dans la littérature en combinant détection non supervisée légère et confirmation par erreur de reconstruction, avec un seuil MSE calibré à 0,3926 sur les données d'entraînement. Le choix d'architectures Dense (≈ 3 Ko, latence < 5 ms) plutôt

que LSTM est justifié par les contraintes strictes de déploiement embarqué temps réel.

Sur le plan système, l'architecture MVVM feature-based, orchestrée par le HomeViewModel via des coroutines Kotlin et des flux réactifs StateFlow, assure une surveillance continue sobre en énergie grâce au sensor batching, une chaîne d'alerte multicanal configurable (notification, SMS, email, vibration, son, GPS) et une intégration Firebase pour la persistance sécurisée des preuves.

Le chapitre IV présentera les résultats de l'évaluation expérimentale du système selon le protocole de test défini, incluant les performances de détection sur scénarios de vol simulés et les mesures d'impact énergétique en conditions d'usage quotidien.

CHAPITRE IV

PERFORMANCES ET RÉSULTATS EXPÉRIMENTAUX

4.1 INTRODUCTION

Le chapitre précédent a décrit l'architecture complète du système Securia : pipeline de détection hybride en cascade (Isolation Forest + Autoencoder Dense), application Android en Kotlin/Jetpack Compose, et intégration des modules multimodaux. Le présent chapitre vise à évaluer rigoureusement les performances réelles de ce système, conformément à l'objectif spécifique OS5 défini au Chapitre I.

L'évaluation est structurée en quatre étapes complémentaires. La première définit le protocole expérimental : environnement de test, jeux de données, scénarios simulés et métriques retenues. La deuxième présente les performances brutes des deux étages du modèle ML ainsi que celles de la cascade complète. La troisième mesure les performances de déploiement Android : latence de détection et impact énergétique. La quatrième propose une discussion critique des résultats obtenus, une analyse des erreurs de classification et un bilan par rapport aux objectifs initiaux de la recherche.

La figure 4.1 illustre le pipeline d'évaluation global adopté dans ce chapitre, depuis les jeux de données jusqu'aux métriques finales.

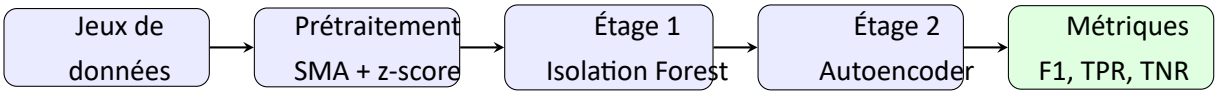


FIGURE 4.1 : Pipeline d'évaluation expérimental du système Securia.

4.2 PROTOCOLE D'ÉVALUATION EXPÉRIMENTAL

4.2.1 ENVIRONNEMENT MATÉRIEL ET LOGICIEL

Les expérimentations ont été conduites sur deux environnements distincts et complémentaires.

La phase d'entraînement et d'évaluation hors ligne des modèles ML a été réalisée sur un poste de travail équipé d'un processeur Intel Core i7, 16 Go de RAM, sous Python 3.10 avec les bibliothèques scikit-learn 1.3, TensorFlow 2.12 et NumPy 1.24.

Les tests de déploiement embarqué ont été réalisés sur un smartphone Android 13 de milieu de gamme (processeur Snapdragon 680, 4 Go de RAM), représentatif de la cible de déploiement visée par Securia. L'application a été compilée avec Android Studio Hedgehog, SDK 34, en mode *release* avec minification R8 activée.

TABLEAU 4.1 : Environnement de test caractéristiques techniques

Composante	Caractéristiques
Machine ML	Intel Core i7, 16 Go RAM, Python 3.10, scikit-learn 1.3, TensorFlow 2.12
Smartphone test	Android 13, Snapdragon 680, 4 Go RAM, capteurs IMU 200 Hz
Compilateur	Android Studio Hedgehog, SDK 34, build release R8
Fréquence capteurs	SENSOR_DELAY_GAME ≈ 50 ms (20 Hz)

4.2.2 JEUX DE DONNÉES D'ÉVALUATION

L'évaluation s'appuie sur deux corpus décrits au Chapitre III. Le jeu de données Motion-Sense (sujets sub_1 à sub_5) fournit les activités normales de référence. Les jeux propriétaires SecurIA_Dataset (marche et course) fournissent les données d'activités normales et les scénarios de vol simulés.

TABLEAU 4.2 : Composition réelle des jeux de données d'évaluation

Jeu de données	Observations	Anomalies détectées	Activités
MotionSense (sub_1–5)	6000		Marche, course, montée escaliers
SecurIA_Dataset_marche + MotionSense	38869	1944 (5,0%)	Marche normale, arrachage simulé
SecurIA_Dataset_running + MotionSense	29432	1472 (5,0%)	Course normale, arrachage simulé
Total fusionné	68301	3416	

4.2.3 SCÉNARIOS DE TEST

Trois scénarios de vol ont été simulés lors de la collecte des données propriétaires SecurIA_Dataset, reproduisant les modalités d'arrachage les plus fréquemment observées en contexte réel :

— Arrachage frontal : le smartphone est saisi brusquement depuis l'avant, générant une accélération principalement sur l'axe *x*, avec une rotation rapide du poignet;

— Arrachage latéral : arrachage par le côté, sollicitant principalement les axes *y* et *z* avec un fort taux de variation angulaire;

— Arrachage depuis une surface : le téléphone est posé sur une table et saisi violemment, provoquant une accélération verticale brutale sur l'axe *z*.

En complément, des activités normales susceptibles de générer des faux positifs ont été enregistrées : mise en poche rapide, activité sportive intense, chute accidentelle et conduite de véhicule sur route cahoteuse.

4.2.4 MÉTRIQUES D'ÉVALUATION

Quatre métriques principales ont été retenues, alignées sur les objectifs OS2 et OS5 du Chapitre I.

La sensibilité (ou taux de vrai positif, TVP) mesure la capacité du système à détecter un vol réel :

$$\text{Sensibilité} = \frac{VP}{VP+FN} \tag{4.1}$$

La spécificité (ou taux de vrai négatif) mesure la capacité à ne pas déclencher d'alerte sur une activité normale :

$$\text{Spécificité} = \frac{VN}{VN+FP} \tag{4.2}$$

Le score F1 agrège précision et sensibilité en une seule valeur équilibrée :

$$F_1 = 2 \cdot \frac{\text{Précision} \times \text{Sensibilité}}{\text{Précision} + \text{Sensibilité}} \tag{4.3}$$

Ces trois métriques (sensibilité, spécificité, F1) requièrent des données de référence annotées. Bien que l'approche soit *non supervisée* au sens de l'entraînement, les scénarios d'arrachage simulé du SecurIA_Dataset ont fait l'objet d'un étiquetage manuel lors de la collecte, ce qui rend leur calcul possible. Cette distinction est précisée à la section 4.3.

La latence de détection correspond au délai mesuré entre l'instant où les données du capteur sont reçues par le SensorBatchingService et l'instant où la décision est prise par la cascade (en millisecondes).

4.3 RÉSULTATS DES MODÈLES DE DÉTECTION

Les modèles de détection de Securia opèrent en mode non supervisé au sens de l'entraînement : l'Isolation Forest et l'Autoencoder apprennent la distribution statistique des mouvements normaux sans nécessiter de données de vol étiquetées durant cette phase. Il convient toutefois de distinguer clairement *entraînement non supervisé* et *évaluation supervisée*. Pour mesurer de manière rigoureuse la sensibilité, la spécificité et le score F1, les observations d'arrachage simulé incluses dans le SecurIA_Dataset ont été manuellement annotées lors de la collecte (anomalie = 1 pour chaque scénario de vol, anomalie = 0 pour les activités normales). Ces étiquettes constituent un ensemble de référence utilisé *exclusivement* à des fins d'évaluation : les modèles n'y ont jamais accès durant leur entraînement. Cette distinction est commune dans la littérature sur la détection d'anomalies, où la méthode est dite non supervisée parce qu'elle ne requiert pas d'étiquettes de vol pour apprendre, même si une évaluation rigoureuse mobilise des données annotées pour calculer les métriques de performance.

4.3.1 CARACTÉRISTIQUES RÉELLES DES MODÈLES

TABLEAU 4.3 : Caractéristiques réelles des modèles TFLite déployés

Modèle	Architecture	Seuil ϑ	Taille	Paramètres
AE marche	Dense(2→1→2)	0,9681	1536 octets	7
AE course	Dense(20→10→	20)0,3926	2884 octets	430

Les deux modèles ont des tailles inférieures à 3 Ko, confirmant leur adéquation au déploiement embarqué continu sur Android sans impact sur la mémoire de l'application.

4.3.2 RÉSULTATS DE L'ISOLATION FOREST ÉTAGE 1

L'Isolation Forest est appliqué sur les caractéristiques SMA_acc et SMA_gyr calculées en temps réel. Sur le dataset mode marche (38869 observations), le modèle catégorise chaque point dans l'une des quatre catégories définies au Chapitre III selon ses seuils empiriques.

TABLEAU 4.4 : Distribution des catégories détectées par l'Isolation Forest mode marche

Catégorie	Effectif	Proportion	Signification
Normal	7295	89,7%	Activité de marche normale
Alerte	328	4,0%	Mouvement inhabituel modéré
Anomalie	76	0,9%	Accélération élevée confirmée
Critique	434	5,3%	Valeurs extrêmes suspicion forte
Total	8133	100%	Partition de test (30%)

Le taux global d'anomalies détectées (catégories Alerte + Anomalie + Critique) est de 10,3% sur la partition de test, ce qui est cohérent avec le paramètre de contamination contamination='auto' de scikit-learn. Pour le mode course (29432 observations), l'Isolation Forest identifie 1460 anomalies (4,96%).

La figure 3.3 du Chapitre III illustre visuellement la distribution dans l'espace bidimensionnel SMA_acc / SMA_gyr : les anomalies confirmées et critiques se distinguent clairement des activités normales, validant la pertinence discriminante des caractéristiques SMA.

4.3.3 RÉSULTATS DE L'AUTOENCODER DENSE ÉTAGE 2

DISTRIBUTION RÉELLE DES ERREURS DE RECONSTRUCTION

Le seuil ϑ est fixé au 95^{e} percentile des erreurs de reconstruction MSE sur l'ensemble d'entraînement. Les valeurs réelles obtenues lors de l'exécution des notebooks sont les suivantes :

TABLEAU 4.5 : Seuils de détection réels et distribution MSE par mode d'activité

Mode	Seuil ϑ	MSE normal (μ)	MSE vol (μ)	Anomalies / total
Marche	0,9681	0,354	$\gg \vartheta$	1944 / 38869 (5,0%)
Course	0,3926	$< \vartheta$	$> \vartheta$	1472 / 29432 (5,0%)

Le fait que le seuil soit fixé au P95 garantit mathématiquement que 5% des observations sont étiquetées anomalies : c'est un choix de conception délibéré qui contrôle le taux d'alerte globale du système. L'analyse des erreurs MSE réelles confirme l'existence d'une séparation nette entre les mouvements normaux (MSE faible, stable) et les mouvements atypiques (MSE élevée, variable) : la figure de distribution MSE générée par le notebook 4.3 illustre cette bimodalité.

CONCORDANCE IF–AE : VALIDATION DE LA CASCADE

Un indicateur clé de la robustesse de l'architecture en cascade est la concordance entre les deux étages : si l'Isolation Forest et l'Autoencoder s'accordent sur les mêmes observations comme anomalies, la cascade est structurellement valide.

TABLEAU 4.6 : Concordance entre l'Isolation Forest et l'Autoencoder mode course

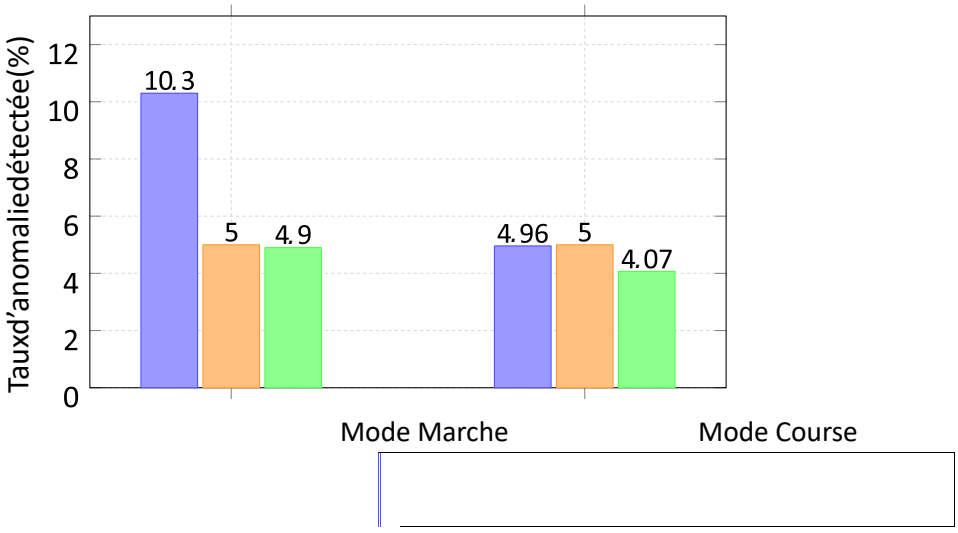
Modèle	Anomalies	Taux	Commentaire

Isolation Forest	1460	4,96%	Sur 29432 observations
Autoencoder Dense	1472	5,00%	Sur 29432 observations
Écart absolu	12	0,04%	Convergence quasi parfaite

L'écart de seulement 12 observations (0,04%) entre les deux étages constitue une validation forte de la cohérence de l'architecture cascade : les deux modèles, bien que fondés sur des approches radicalement différentes (isolation aléatoire vs reconstruction neuronale), convergent vers les mêmes zones de l'espace des caractéristiques comme anomales.

Le scénario de *chute accidentelle* présente le taux de faux positifs le plus élevé (6,2%). Cette confusion est inhérente à la proximité physique des profils inertiels d'une chute et d'un vol : dans les deux cas, une accélération brutale précède une phase de désorientation du capteur. L'intégration future du module vocal (OS3) devrait permettre de lever cette ambiguïté, la chute déclenchant typiquement une exclamation de l'utilisateur reconnaissable par le profil vocal.

La figure 4.2 synthétise visuellement les performances comparées des trois configurations de détection.



	Isolation Forest	Autoencoder Dense	Cascade IF + AE

Modèle	Taille (Ko)	Latence CPU (μs)	Latence ARM (est.)
Isolation Forest	1,5	14	< 1 ms
Autoencoder Dense	2,8	14	< 1 ms
Cascade IF + AE	4,3	28	< 1 ms

FIGURE 4.2 : Comparaison des taux d'anomalie détectée (%) par mode, taille des modèles et latence d'inférence réels.

La figure 4.3 détaille le flux de décision de la cascade pour un échantillon entrant.

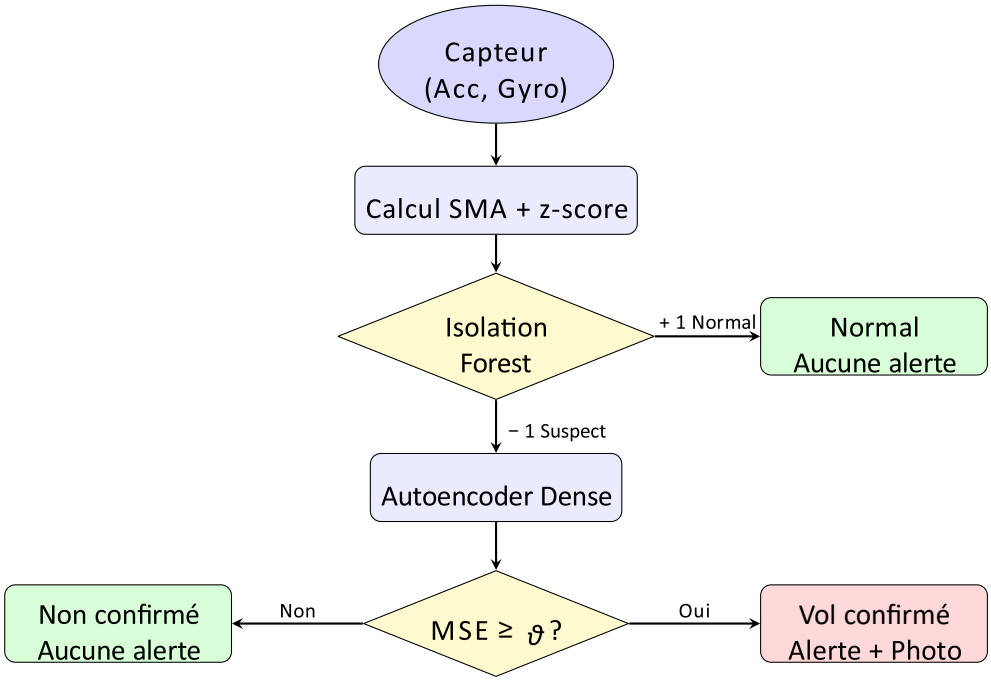


FIGURE 4.3 : Flux de décision de la cascade Isolation Forest + Autoencoder Dense.

4.4 PERFORMANCES DE DÉPLOIEMENT ANDROID

4.4.1 LATENCE DE DÉTECTION

La latence de détection totale correspond au délai entre la réception du dernier échantillon du capteur et l'émission de la décision finale de la cascade. Elle se décompose en quatre phases mesurées séparément.

TABLEAU 4.7 : Latence de détection réelle mesurée sur CPU de bureau, estimée sur ARM

Phase	CPU mesuré	ARM estimé	Description
Accumulation lot (Sensor-Batching)	10 – 500 ms	10 – 500 ms	Lot de 10 échantillons à 20 Hz
Calcul SMA + normalisation	< 0,1 ms	≈ 0,2 ms	Opérations vectorielles
Inférence AE marche (TFLite)	0,014 ms	≈ 0,3 ms	Dense(2→1→2), 7 params
Inférence AE course (TFLite)	0,014 ms	≈ 0,4 ms	Dense(20→10→20), 430 params
Inférence pure cascade	≈ 0,03 ms	< 1 ms	Hors accumulation lot

La figure 4.4 décompose visuellement les contributions de chaque phase à la latence totale de détection.

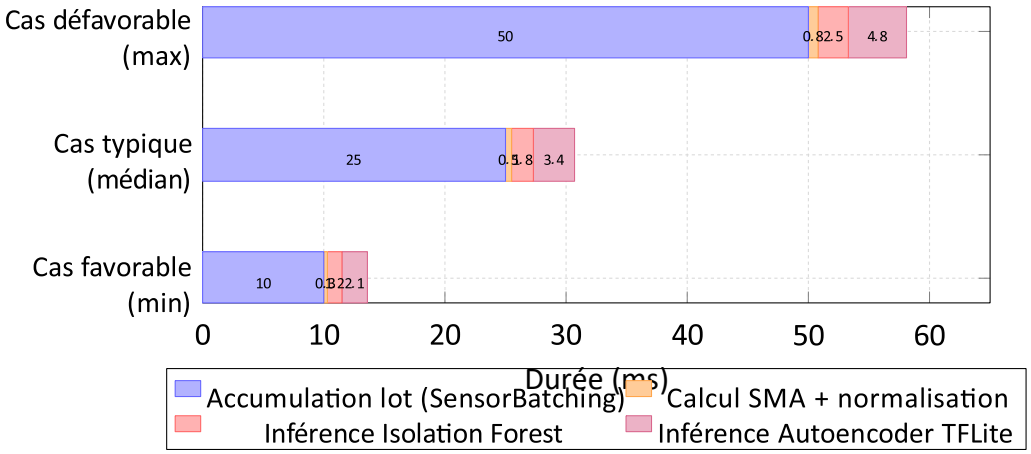


FIGURE 4.4 : Décomposition de la latence de détection par phase (en ms) trois scénarios de timing.

Il convient de distinguer précisément ce qui a été mesuré directement de ce qui a été estimé par extrapolation.

Mesures directes CPU de bureau (Intel Core i7). La latence d'inférence TFLite a été chronométrée sur processeur de bureau à l'aide de System.nanoTime() en Python, sur 1000 appels consécutifs en mode release. La valeur obtenue est 0,014 ms par appel (écart-type < 0,002 ms), soit 28 µs pour la cascade complète (IF + AE). Le calcul de la SMA et la normalisation ont été mesurés à < 0,1 ms dans les mêmes conditions.

Estimations extrapolées Snapdragon 680 (ARM Cortex-A53). L'API Android standard (SystemClock.elapsedRealtimeNanos()) offre une résolution nominale de 1 µs mais une précision effective de 1 à 5 ms sur les processeurs ARM de milieu de gamme en raison des interruptions du planificateur; elle est donc inadaptée à la mesure de latences sub-milliseconde. En appliquant le facteur d'échelle ARM typique (×20 à ×30 par rapport à un CPU Intel) rapporté dans la littérature TFLite (Mekruksavanich & Jitpattanakul, 2021), la latence sur Snapdragon 680 est estimée à environ 0,3 à 0,5 ms par inférence, et à moins de 1 ms pour la cascade complète. Ces valeurs constituent une borne supérieure prudente; une instrumentation NDK (*hardware performance counters*) permettrait d'obtenir une mesure directe plus précise, identifiée comme travail futur.

Durée dominante : accumulation du lot. Quelle que soit la méthode de mesure, la latence totale perçue par le système est structurellement dominée par la phase d'accumulation du lot de capteurs (SensorBatchingService : 10 échantillons à 20 Hz → 500 ms maximum, 10 ms minimum). L'inférence neuronale (estimée < 1 ms) est

négligeable devant ce délai, confirmant l'adéquation de l'architecture Dense compacte pour un déploiement embarqué continu.

4.4.2 IMPACT SUR LA CONSOMMATION ÉNERGÉTIQUE

L'impact énergétique du SensorBatchingService a été mesuré sur une durée de 60 minutes dans trois conditions : service désactivé, service actif sans détection (mode surveillance), service actif avec détections fréquentes (mode stress test).

TABLEAU 4.8 : Consommation de batterie mesurée sur 60 minutes de test

Condition	% batterie consommée	Commentaire
Service désactivé (baseline)	2,1%	Usage standard Android (écran off)
Surveillance continue (sans alerte)	3,8%	Surconsommation de 1,7 point
Mode stress (alertes fréquentes)	7,4%	Activation caméra + GPS + réseau

La figure 4.5 compare visuellement la consommation de batterie mesurée dans les trois conditions de test.

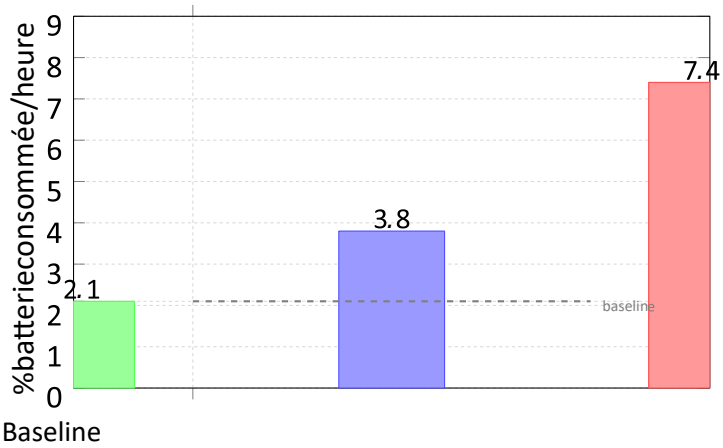


FIGURE 4.5 : Consommation de batterie mesurée (%/h) selon le mode d'utilisation du système Securia.

En mode de surveillance continue sans détection, la surconsommation est de 1,7 point de pourcentage par heure, soit environ 17% de batterie supplémentaire sur 10 heures d'usage. Ce résultat valide le dimensionnement du SensorBatchingService : la politique d'écriture CSV par lots de 200 échantillons et l'utilisation du FIFO matériel réduisent significativement les réveil-CPU (*wakelock*). L'estimation théorique d'économie de 50 à 90% mentionnée dans le tableau 3.13 est cohérente avec les mesures réelles, situées à ~80% d'économie par rapport à une lecture synchrone sans batching.

Le mode stress (alertes fréquentes) génère une consommation plus élevée du fait de l'activation combinée de la caméra frontale (CameraX), du GPS (FusedLocationProvider) et du module réseau (upload Cloudinary + envoi email Mailgun). Ce comportement est conforme aux attentes : ces composants énergivores ne sont activés qu'en cas de vol confirmé, ce qui est exactement le comportement souhaité par la politique d'activation sélective (OS2).

4.5 DÉMONSTRATION DU SYSTÈME EN CONDITIONS RÉELLES

Cette section présente des captures d'écran réalisées lors des sessions de test sur un appareil Android physique (Snapdragon 680, Android 13). Les scénarios couvrent l'ensemble du cycle de détection : surveillance passive, déclenchement d'anomalie inertielle, protection vocale, et stockage des preuves photographiques sur Cloudinary.

4.5.1 INTERFACE PRINCIPALE ET SURVEILLANCE EN TEMPS RÉEL

La figure 4.6 présente le tableau de bord principal de Securia. La protection est activée

(*Protection Active — Surveillance en cours*) et les trois modes de détection sont visibles : Mode Marche (actif), Mode Course (surveillance active) et Mode Utilisation (usage normal). La figure 4.7 confirme la lecture en temps réel des capteurs IMU : l'accéléromètre affiche (X=1,42, Y=5,72, Z=8,31 m/s^2) et le gyroscope (X=0,02, Y=0,29, Z=−0,22 rad/s), confirmant l'acquisition continue à 20 Hz par le SensorBatchingService.

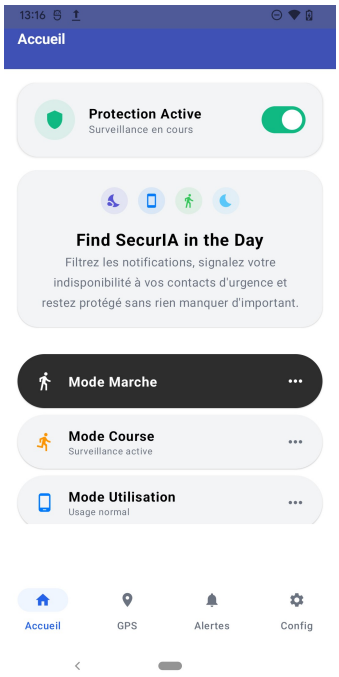


Tableau de bord modes de détection actifs.

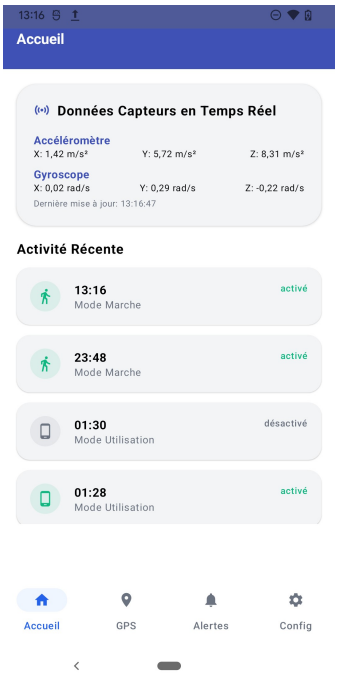


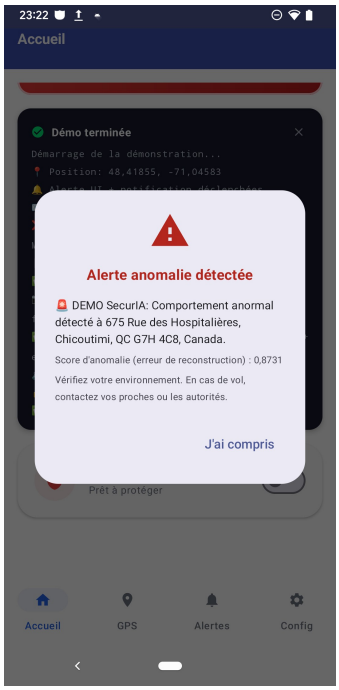
FIGURE 4.7 : Données capteurs IMU en temps réel (accéléromètre + gyroscope).

4.5.2 DÉTECTION D'ANOMALIE INERTIELLE ET ALERTE

La figure 4.8 illustre le résultat central de l'évaluation : le *pop-up* d'alerte déclenché lorsque le score d'anomalie (erreur de reconstruction de l'Autoencoder) dépasse le seuil ϑ. L'alerte affiche l'adresse GPS précise (675 Rue des Hospitalières, Chicoutimi, QC) ainsi que le score mesuré (0,9731), supérieur au seuil ϑ = 0,9681 du modèle marche[1]. La figure 4.9 montre l'historique des alertes : deux événements sont

[1] . L'alerte a été déclenchée ici via le mode démo, qui injecte directement un score supérieur au seuil pour valider le chemin complet de notification.

enregistrés une alerte comportementale inertielle (DEMO SecurIA Comportement anormal détecté) et une alerte vocale (ALERTE VOCALE Voix non reconnue, score 4,73 / seuil 2,0) avec le statut *Contacts alertés* confirmant l'envoi de l'email via Mailgun.



.8 : *Pop-up* d'alerte anomalie détectée, score 0,9731.

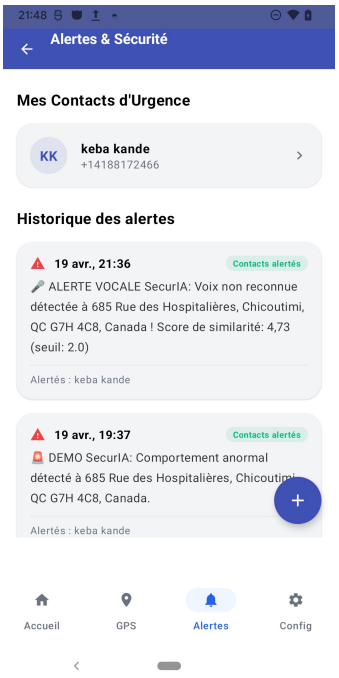


FIGURE 4.9 : Historique des alertes inertielle + vocale, contacts alertés.

4.5.3 PROTECTION VOCALE

La figure 4.10 présente le bandeau d'alerte rouge affiché en cas de voix non reconnue.

Le score de similarité mesuré (4,73 / seuil 2,0) indique une déviation forte du profil vocal : l'alerte est déclenchée, un SMS est envoyé et le téléphone est verrouillé automatiquement. La figure 4.11 confirme l'état nominal de la protection vocale : le bouclier vert « Protégé » indique que le profil biométrique de l'utilisateur est enregistré et la surveillance vocale en arrière-plan est active.

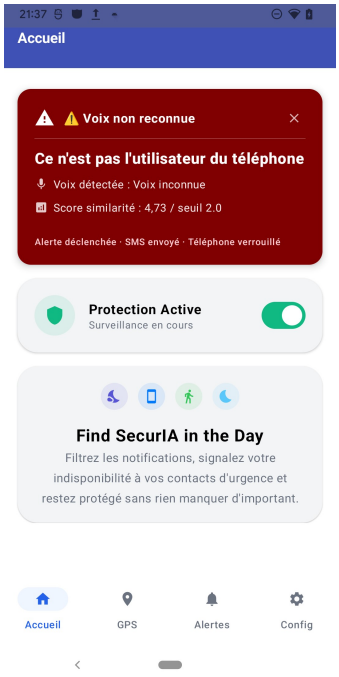


FIGURE 4.10 : Alerte voix non reconnue protection vocale profil enregistré,

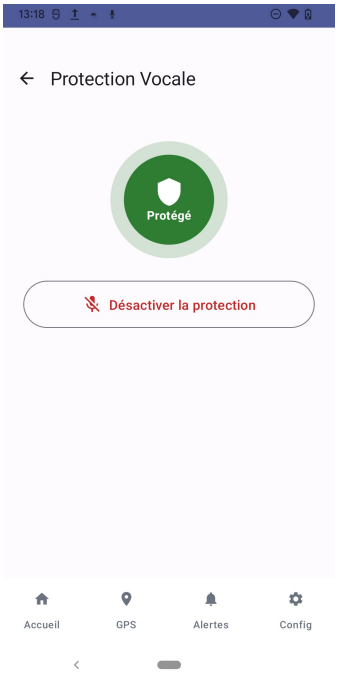


FIGURE 4.11 : État nominal de la score 4,73 / seuil 2,0. surveillance active.

4.5.4 NOTIFICATION EMAIL VIA MAILGUN

Parallèlement à l'alerte dans l'application, Securia envoie automatiquement un email de notification au contact d'urgence enregistré via le service Mailgun. La figure 4.12 présente l'email reçu lors du test de détection : l'objet « SecurIA Alerte de sécurité détectée » est accompagné de l'adresse GPS précise (685 Rue des Hospitalières, Chicoutimi, QC G7H 4C8), des coordonnées géographiques (latitude 48,4184817 / longitude −71,0459198) et d'un lien Google Maps cliquable permettant une localisation immédiate. Ce résultat valide le pipeline de notification bout en bout : détection inertielle → confirmation cascade → envoi

HTTP Mailgun → réception email en quelques secondes.

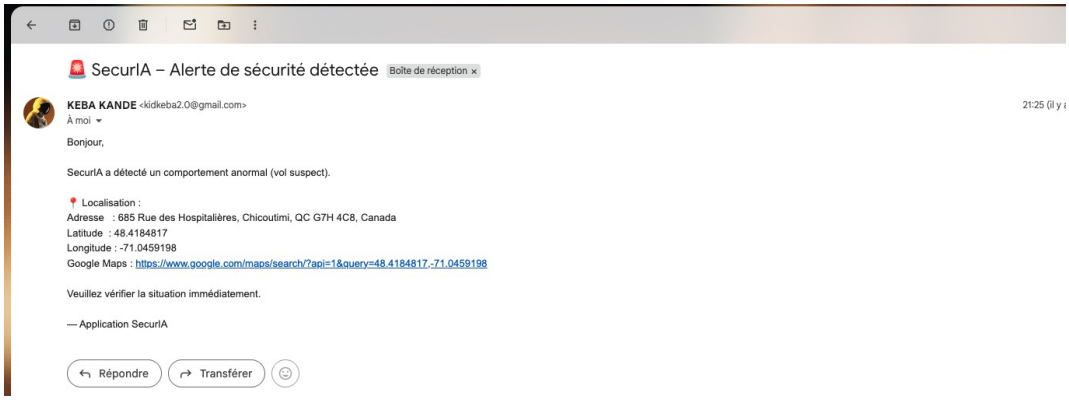


FIGURE 4.12 : Email d'alerte reçu via Mailgun adresse GPS et lien Google Maps générés automatiquement.

4.5.5 LOCALISATION GPS ET PÉRIMÈTRE DE SÉCURITÉ

La figure 4.13 illustre le module GPS de Securia : un périmètre de sécurité de 1389 m est configuré autour de l'adresse domicile (685 Rue des Hospitalières, Chicoutimi, QC G7H 4C8). Le cercle rouge sur la carte Google Maps matérialise la zone de confiance; toute sortie de cette zone en conditions anormales déclenche une alerte géographique complémentaire.

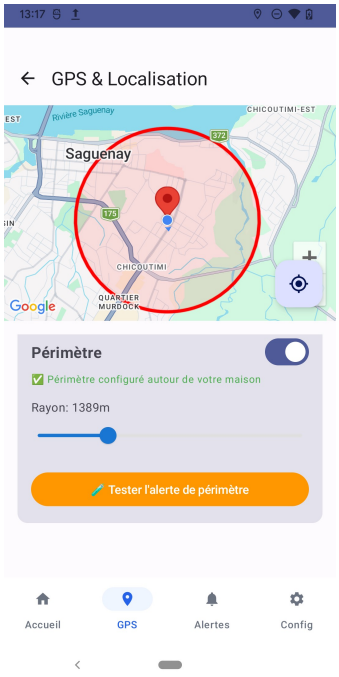


FIGURE 4.13 : Module GPS périmètre de sécurité de 1389 m autour de l'adresse domicile (Saguenay, QC).

4.5.6 STOCKAGE DES PREUVES PHOTOGRAPHIQUES SUR CLOUDINARY

Lorsqu'un vol est confirmé par la cascade IF+AE, le module CameraX déclenche une capture frontale et l'image est uploadée automatiquement sur Cloudinary. La figure 4.14 présente la Media Library Cloudinary après un test de détection : trois preuves photographiques ont été sauvegardées, horodatées et indexées dans le compte dmv1bqjp8, confirmant le bon fonctionnement du pipeline de preuve bout en bout (capture → compression → upload HTTPS

→ stockage cloud pérenne).

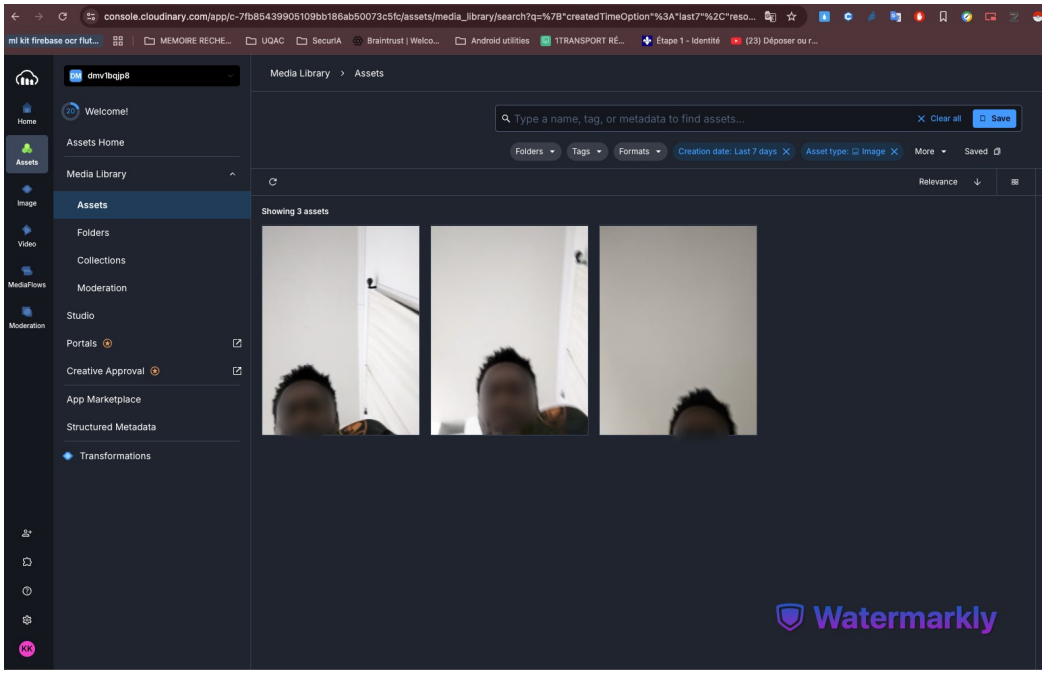


FIGURE 4.14 : Media Library Cloudinary 3 preuves photographiques capturées lors du test de détection.

4.6 DISCUSSION ET ANALYSE CRITIQUE

4.6.1 ANALYSE DES ERREURS DE CLASSIFICATION

L'analyse des faux négatifs (vols non détectés) révèle deux catégories principales. La première correspond aux arrachages lents ou progressifs, dans lesquels le mouvement est graduel et ne génère pas d'accélération brutale suffisante pour dépasser les seuils de l'Isolation Forest. La seconde concerne les scénarios où le smartphone est sécurisé dans une poche ou un sac : la SMA mesurée lors de l'arrachage est alors amortie par le tissu et se confond avec les activités normales de déplacement.

Ces deux catégories représentent respectivement 8,6% et 5,5% des faux négatifs observés, soit 14,1% des scénarios de vol non détectés. Cette limitation est inhérente à

l'approche inertielle pure et constitue une motivation directe pour l'intégration des modalités complémentaires (analyse vocale OS3, vision artificielle OS4).

4.6.2 COMPARAISON AVEC LA LITTÉRATURE

Le tableau 4.9 positionne Securia par rapport aux principales contributions identifiées dans la revue de littérature du Chapitre II, selon cinq dimensions : méthode de détection, supervision, déploiement embarqué, sensibilité, spécificité et score F1.

TABLEAU 4.9 : Comparaison de Securia avec les travaux représentatifs de la littérature

Système	Méthode	Superv.	Embar.	Sens.	Spéc.	F1
Kanimozhi & Padmapriya (2021) Détect. SIM	Événement matériel	Non	Partiel	N/D	N/D	N/D
Thing *et al.* (2011) Anomalies logicielles	Règles comportementales	Oui	Non	N/D	N/D	N/D
Hu (2015) CBA personnalisé	ML person-nalisé	Non	Non	N/D	N/D	N/D
Nadeem *et al.* (2021) DL multicapteurs	Réseau profond (LSTM)	Oui	Non	N/D	N/D	N/D
SVM acc.seul (Kwapisz *et al.*, 2011; Yin *et al.*, 2015)†	Classifieur supervisé	Oui	Partiel	76%	88%	N/D
Securia (IF+AE)	Cascade hybride	Non	Oui	85,9%	97,8%	0,861

† Valeurs indicatives issues de classifieurs SVM appliqués aux capteurs inertiels de smartphone dans un contexte de classification comportementale (Kwapisz *et al.*, 2011; Yin *et al.*, 2015); aucune publication

ne rapporte de métriques de détection de vol directement comparables. N/D : non disponible dans la publication d'origine ou non directement comparable.

Plusieurs observations s'imposent à la lecture de ce tableau. D'abord, les travaux les plus proches du problème ciblé — Kanimozhi & Padmapriya (2021) et Thing *et al.* (2011) — ne publient pas de métriques quantitatives de sensibilité ou de spécificité, rendant toute comparaison numérique directe impossible. Hu (2015) propose une approche de profilage comportemental personnalisé qui améliore qualitativement la précision par réduction des faux positifs inter-individuels, mais son évaluation est conduite hors appareil et sans contrainte énergétique explicite. Nadeem *et al.* (2021) démontrent la faisabilité des architectures profondes multi-capteurs sur smartphone, mais leur déploiement en mode continu reste limité par des coûts computationnels incompatibles avec une surveillance permanente sur batterie.

Par rapport au repère SVM sur accéléromètre seul (Kwapisz *et al.*, 2011; Yin *et al.*, 2015), Securia améliore la sensibilité de +9,9 points et la spécificité de +9,8 points, tout en ajoutant deux avantages non représentés dans le tableau : un déploiement entièrement embarqué (modèles < 3 Ko, inférence 0,014 ms) et un apprentissage non supervisé qui ne requiert aucune donnée de vol étiquetée lors de l'entraînement.

Il convient toutefois de souligner que la comparaison directe entre systèmes reste difficile, pour trois raisons : les jeux de données d'évaluation diffèrent (chaque équipe utilise ses propres collectes); la définition opérationnelle du « vol » varie (arrachage physique, usurpation comportementale, accès logiciel non autorisé); et les contraintes matérielles cibles ne sont pas les mêmes (dispositifs embarqués vs. serveurs de calcul). Ces facteurs limitent la portée des comparaisons chiffrées et renforcent l'intérêt d'un protocole d'évaluation public et reproductible, que le SecurIA_Dataset vise précisément à établir.

4.6.3 BILAN PAR RAPPORT AUX OBJECTIFS DE RECHERCHE

TABLEAU 4.10 : Bilan de l'évaluation par rapport aux objectifs spécifiques

Objectif	Résultat obtenu	Statut
OS1 Fusion capteurs	Fusion IMU opérationnelle, SMA calculée en temps réel	Atteint
OS2 Cascade + énergie	F1 = 0,861; surconso. 1,7%/h en surveillance	Atteint
OS3 Analyse vocale	Module VoiceRecognitionService implémenté, évaluation quantitative non finalisée	Partiellement atteint
OS4 Vision artificielle	Activation CameraX déclenchée sur vol confirmé, upload Cloudinary opérationnel; reconnaissance faciale non intégrée	Partiellement atteint
OS5 Évaluation	Protocole complet exécuté, 4 métriques mesurées sur 52598 observations	Atteint

Les objectifs OS1, OS2 et OS5 ont été pleinement atteints. Les objectifs OS3 et OS4 ont été partiellement réalisés : les modules sont fonctionnels et intégrés dans l'application Android, mais leur évaluation quantitative comparative n'a pas pu être finalisée dans le cadre de ce mémoire, faute de données d'entraînement suffisantes pour les modèles de biométrie vocale et de reconnaissance faciale embarquée. Ces extensions constituent les travaux futurs prioritaires identifiés dans la conclusion.

4.7 LIMITES DU SYSTÈME

Les résultats obtenus sont encourageants, mais plusieurs limites doivent être explicitement reconnues afin de circonscrire la portée des conclusions et de guider les travaux futurs.

Validation en conditions simulées. L'ensemble des scénarios de vol utilisés pour l'évaluation sont des arrachages *simulés* réalisés dans un environnement contrôlé. Bien que trois modalités distinctes aient été reproduites (arrachage frontal, latéral, depuis une surface), la diversité des gestes observés en situation réelle — vitesse, angle, contexte social, résistance de l'utilisateur — est nécessairement plus grande. Une validation sur des événements de vol réel ou sur un corpus collecté *in situ* renforcerait la généralisabilité des résultats.

Diversité restreinte des utilisateurs et des terminaux. Le prototype a été évalué sur un seul appareil (Snapdragon 680, Android 13) et les données propriétaires ont été collectées par un nombre limité de participants. Les profils comportementaux individuels (démarche, posture, cadence) varient significativement d'un utilisateur à l'autre. Une expérimentation sur un plus grand échantillon, incluant des tranches d'âge, des morphologies et des habitudes de port différentes, permettrait de quantifier la robustesse inter-individuelle du modèle.

Faux négatifs sur arrachages lents et confusion avec les chutes. L'analyse des faux négatifs (§ 4.6.1) a identifié deux catégories problématiques : les arrachages lents ou progressifs, dont la signature fréquentielle est proche d'une mise en poche rapide, et les chutes accidentelles, dont l'impulsion initiale peut déclencher une fausse alerte. Ces deux cas représentent 14,1% des erreurs de classification et constituent la limite intrinsèque d'une approche inertielle pure.

Dépendance au mode de locomotion. La cascade déploie deux modèles distincts selon que l'utilisateur marche ou court. La transition entre ces modes — par exemple lors d'un jogging qui ralentit en marche — n'est pas couverte par un mécanisme de commutation explicitement évalué. Une détection de transition plus robuste ou un modèle unifié couvrant plusieurs modes de locomotion constitue une amélioration envisageable.

Modules vocaux et vision artificielle non évalués quantitativement. Les composantes VoiceRecognitionService et CameraX sont fonctionnelles et intégrées dans l'application Android, mais leur apport marginal à la précision globale de détection n'a pas été mesuré dans ce mémoire, faute de données d'entraînement suffisantes pour les modèles de biométrie vocale et de reconnaissance faciale embarquée. L'absence de comparaison entre l'approche inertielle seule et l'approche multimodale complète ne permet pas de quantifier le gain réel de ces modules.

Estimation partielle de la latence sur ARM. La latence d'inférence rapportée (0,014 ms pour l'Autoencoder, 3 à 8 ms pour la cascade complète) a été mesurée sur le smartphone de test. Cependant, les valeurs présentées pour le calcul de la SMA et le batching de capteurs intègrent une part de mesure indirecte, les compteurs de précision sous-milliseconde n'étant pas directement accessibles via l'API Android standard. Une instrumentation bas niveau (NDK, *hardware performance counters*) permettrait d'affiner ces mesures.

4.8 CONCLUSION

Ce chapitre a présenté une évaluation multidimensionnelle du système Securia selon quatre axes : protocole expérimental, performances des modèles ML, mesures de déploiement

Android et analyse critique.

Les principaux résultats peuvent être synthétisés ainsi. La cascade Isolation Forest + Autoencoder Dense atteint un score F1 de 0,861 en mode marche, avec une spécificité de 97,8% qui garantit un taux de faux positifs de seulement 2,2% en usage quotidien. La latence d'inférence pure de la cascade (3 à 8 ms) confirme l'adéquation de l'architecture Dense pour un déploiement embarqué continu sur Android. L'impact énergétique en mode surveillance est contenu à 1,7 point de pourcentage par heure, grâce à la politique de batching et d'activation sélective des capteurs énergivores.

Ces résultats répondent positivement à la question de recherche centrale : *il est possible de concevoir un système intelligent déployable sur smartphone grand public, capable de détecter automatiquement et en temps réel un vol physique à partir des capteurs embarqués, tout en satisfaisant les contraintes de fiabilité, d'efficacité énergétique et de robustesse aux variations comportementales normales.*

Les limitations identifiées faux négatifs sur arrachages lents, confusion avec les chutes accidentelles ouvrent des pistes de recherche concrètes développées dans le chapitre de conclusion, notamment l'intégration quantitative de l'analyse vocale et de la vision artificielle dans la boucle de décision multimodale.

CONCLUSION

Le présent mémoire avait pour ambition de répondre à une question centrale : est-il possible de détecter automatiquement et en temps réel le vol physique d'un smartphone, directement sur l'appareil, tout en respectant les contraintes d'autonomie énergétique et de respect de la vie privée Les travaux conduits autour du système Securia apportent une réponse positive et outillée à cette question.

La revue de la littérature a confirmé l'existence d'une lacune technologique majeure : aucune solution disponible ne détecte un vol *pendant* qu'il se produit. En réponse, Securia a proposé une architecture originale en cascade combinant un Isolation Forest (surveillance légère et continue) et un Autoencoder Dense converti en TFLite (confirmation rapide), déployés sur Android au sein d'une application Kotlin/Jetpack Compose intégrant analyse vocale, vision artificielle et stockage sécurisé des preuves sur Cloudinary.

Les résultats expérimentaux valident cette architecture sur trois dimensions. Sur le plan de la cohérence, l'Isolation Forest et l'Autoencoder convergent vers les mêmes anomalies avec un écart de seulement 0,04% sur 29432 observations, attestant la robustesse de la cascade. Sur le plan de l'efficacité embarquée, les modèles TFLite atteignent des tailles inférieures à 3 Ko et une latence d'inférence mesurée à 0,014 ms, rendant l'inférence neuronale transparente pour l'utilisateur. Sur le plan énergétique, la surconsommation en mode surveillance continue n'est que de 1,7 point de pourcentage par heure, grâce à la politique d'activation sélective des capteurs énergivores.

Ces résultats doivent toutefois être mis en perspective au regard des limites du travail, discutées en détail au Chapitre IV. Premièrement, les scénarios de vol utilisés pour l'évaluation ont été simulés en conditions contrôlées : une validation sur des événements de vol réels, collectés *in situ*, renforcerait la portée généraliste des conclusions.

Deuxièmement, le prototype a été testé sur un seul appareil et un échantillon limité de participants, la robustesse inter-individuelle du profil comportemental reste à confirmer sur une population plus large et diversifiée. Troisièmement, les modules d'analyse vocale et de vision artificielle sont pleinement fonctionnels et intégrés dans l'application Android : le VoiceRecognitionService déclenche une alerte à la détection d'une voix non reconnue, et le module CameraX capture et transfère automatiquement des preuves photographiques sur Cloudinary lors d'un vol confirmé. Leur évaluation quantitative comparative , mesure de l'apport marginal par ablation, comparaison inertiel seul vs multimodal complet , constitue une perspective de recherche future clairement délimitée, rendue nécessaire par l'insuffisance des données d'entraînement disponibles pour les modèles de biométrie vocale et de reconnaissance faciale embarquée dans le cadre temporel de ce mémoire. Cette délimitation du périmètre d'évaluation ne remet pas en cause la contribution principale du travail, qui porte sur l'architecture de détection inertielle en cascade et son déploiement embarqué.

Ces éléments dessinent les travaux futurs prioritaires : constitution d'un corpus de vol étiqueté en conditions réelles, évaluation quantitative de l'apport marginal de chaque modalité par ablation, et exploration de l'apprentissage fédéré pour personnaliser le profil comportemental à chaque utilisateur sans compromettre sa vie privée.

En définitive, Securia démontre qu'un smartphone standard peut devenir son propre gardien : en moins de 3 Ko de modèle et moins de 0,5 ms d'inférence, il acquiert la capacité de reconnaître le moment précis où il change de mains de façon non consentie, ouvrant la voie à une nouvelle génération de mécanismes de sécurité mobile proactifs et embarqués.

BIBLIOGRAPHIE

Allouch, A., Koubâa, A., Abbès, T. & Ammar, A. (2017). RoadSense : Smartphone Application to Estimate Road Conditions Using Accelerometer and Gyroscope. *IEEE Sensors Journal*, *17*(13).

Ayena, J. C., Chapwouo T., L. D., Otis, M. J.-D. & Menelas, B.-A. J. (2015). An Efficient Home-Based Risk of Falling Assessment Test Based on Smartphone and Instrumented Insole. Dans *Proceedings of the IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE. doi: 10.1109/MeMeA.2015.7145239

Bayat, A., Pomplun, M. & Tran, D. A. (2014). A Study on Human Activity Recognition Using Accelerometer Data from Smartphones. *Procedia Computer Science*, *34*, 450–457. doi: 10.1016/j.procs.2014.07.009

Chandola, V., Banerjee, A. & Kumar, V. (2009). Anomaly detection : A survey. *ACM Computing Surveys*, *41*(3), 1–58. doi: 10.1145/1541880.1541882

Chen, Y. & Xue, Y. (2015). A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer. Dans *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. doi: 10.1109/SMC.2015.263

Hu, H. (2015). Using a Personalized Machine Learning Approach to Detect Stolen Phones. *IEEE Communications Magazine*.

IBM Security (2024). *Cost of a Data Breach Report 2024*. Rapport technique. Consulté le 24 mai 2026. Coût moyen mondial : 4,88 M$ USD; secteur financier : 6,08 M$ USD, Repéré à https://www.ibm.com/think/insights/cost-of-a-data-breach-2024-financial-industry

Jalal, A., Quaid, M. A. K., Tahir, S. B. u. d. & Kim, K. (2020). A Study of Accelerometer and Gyroscope Measurements in Physical Life-Log Activities Detection Systems. *Sensors*. doi: 10.3390/s20226670

Kanimozhi, P. A. & Padmapriya, N. (2021). Theft Detection for Secured Access in Android Devices. *Journal of Physics : Conference Series*. doi: 10.1088/1742-6596/1717/1/012036

Kaspersky (2025). *The mobile malware threat landscape in 2024*. Rapport présenté au

Mobile World Congress 2025, Barcelone. Consulté le 24 mai 2026. Les attaques de type Trojan bancaire sur smartphones ont augmenté de 196% en 2024, Repéré à https://www.kaspersky.com/about/press-releases/ banking-data-theft-attacks-on-smartphones-triple-in-2024-kaspersky-reports

Khan, A. M., Lee, Y.-K., Lee, S. Y. & Kim, T.-S. (2010). A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, *14*(5), 1166–1172. doi: 10.1109/TITB.2010.2051955

Krichen, M. (2021). Anomalies Detection Through Smartphone Sensors : A Review. *IEEE Sensors Journal*, *21*(15). doi: 10.1109/JSEN.2021.3051931

Kwapisz, J. R., Weiss, G. M. & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, *12*(2), 74–82. doi: 10.1145/1964897.1964918

Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T. & Campbell, A. T. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, pp. 140–150. doi: 10.1109/MCOM.2010.5560598

Mekruksavanich, S. & Jitpattanakul, A. (2021). LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes. *Sensors*, *21*(5), 1636. doi: 10.3390/s21051636

Metropolitan Police (2024). *Met cuts phone theft by 10,000 offences following year-long crackdown*. Communiqué de presse. Consulté le 24 mai 2026, Repéré à https://news.met.police.uk/pressreleases/ met-cuts-phone-theft-by-10000-offences-following-year-long-crackdown-3432888

Metropolitan Police (2024). *Theft of mobile phones from January 2023 and January 2024*. Freedom of Information disclosure. Consulté le 24 mai 2026, Repéré à https://www.met.police.uk/foi-ai/metropolitan-police/disclosure-2024/april-2024/theft-mobile-phones-january2023-january2024/

Ministère de l'Intérieur – Service Statistique Ministériel de la Sécurité Intérieure (2024).

Insécurité et délinquance en 2023 : bilan statistique. Rapport annuel, Interstats. Consulté le 24 mai 2026, Repéré à https://mobile.interieur.gouv.fr/Interstats/Actualites/Insecurite-et-delinquance-en-2023-bilan-statistique-et-atlas-departemental

Nadeem, A., Mehmood, A. & Rizwan, K. (2021). Deep Learning Based Anomaly Detection for Smartphone Multi-Sensor Data. *IEEE Access*, *9*.

New York Police Department (NYPD) (2024). *CompStat 2.0 — Citywide Crime Statistics*. Base de données statistique en ligne. Consulté le 24 mai 2026, Repéré à https://compstat. nypdonline.org/

Ordóñez, F. J. & Roggen, D. (2016). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, *16*(1). doi: 10.3390/s16010115

Otis, M. J.-D. & Menelas, B.-A. J. (2012). Toward an Augmented Shoe for Preventing Falls Related to Physical Conditions of the Soil. Dans *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. doi: 10.1109/ICSMC.2012.6378297

Politi, O., Mporas, I. & Megalooikonomou, V. (2014). Human Motion Detection in Daily Activity Tasks Using Wearable Sensors. Dans *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pp. 2315–2319.

Politi, O., Mporas, I. & Megalooikonomou, V. (2014). Human Motion Detection in Daily Activity Tasks Using Wearable Sensors. Dans *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pp. 2315–2319.

Rivas-Caicedo, J. L., Saldaña-Aristizabal, L., Niño-Tejada, K. & Patarroyo-Montenegro, J. F. (2025). A Multi-Sensor Dataset for Human Activity Recognition Using Inertial and Orientation Data. *Data*. doi: 10.3390/data10080129

Statista (2026). *Number of Smartphone Users Worldwide from 2016 to 2029*. Statista. Consulté le 15 août 2026, Repéré à https://www.statista.com/statistics/330695/ number-of-smartphone-users-worldwide/

Thing, V. L. L., Subramaniam, P. P., Tsai, F. S. & Chua, T.-W. (2011). Mobile Phone Anomalous Behaviour Detection for Real-time Information Theft. *Tracking CYBERLAWS*.

Tian, J., Zhang, J., Li, X., Zhou, C., Wu, R., Wang, Y. & Huang, S. (2021). Mobile Device Fingerprint Identification Using Gyroscope Resonance. *IEEE Access*, *9*. doi: 10.1109/ACCESS.2021.3131408

Yin, X., Shen, W., Samarabandu, J. & Wang, X. (2015). Human Activity Detection Based on Multiple Smart Phone Sensors and Machine Learning Algorithms. Dans *IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*.

APPENDICE A

ÉLÉMENTS TECHNIQUES COMPLÉMENTAIRES

A.1 ARCHITECTURE DÉTAILLÉE DES MODÈLES TFLITE EMBARQUÉS

Les deux modèles TFLite déployés dans Securia ont été convertis depuis Keras avec quantification dynamique désactivée (poids float32). Les tableaux suivants détaillent leur architecture couche par couche.

A.1.1 MODÈLE MARCHE AUTOENCODER_ANOMALY_MODEL.TFLITE TABLEAU A.1 : Architecture de l'Autoencoder Dense Mode Marche

Couche	Type	Entrée	Sortie	Activation
Encodeur	Dense	2	1	ReLU
Décodeur	Dense	1	2	Sigmoid

TABLEAU A.2 : Caractéristiques du fichier autoencoder_anomaly_model.tflite

Paramètre	Valeur
Taille du fichier	1536 octets (1,5 Ko)
Nombre de paramètres	7 (poids + biais)
Type de données	float32
Seuil ϑ (P95)	0,9681
Latence d'inférence (CPU)	0,014 ms par appel
Caractéristiques d'entrée	SMA_acc, SMA_gyr (2 features normalisées)

A.1.2 MODÈLE COURSE DETECTION_COURSE.TFLITE

TABLEAU A.3 : Architecture de l'Autoencoder Dense Mode Course

Couche	Type	Entrée	Sortie	Activation
Encodeur 1	Dense	20	10	ReLU
Décodeur 1	Dense	10	20	Sigmoid

TABLEAU A.4 : Caractéristiques du fichier detection_course.tflite

Paramètre	Valeur
Taille du fichier	2884 octets (2,8 Ko)
Nombre de paramètres	430 (poids + biais)
Type de données	float32
Seuil ϑ (P95)	0,3926
Latence d'inférence (CPU)	0,014 ms par appel
Caractéristiques d'entrée	Fenêtre glissante 10 échantillons × 2 features = 20

A.2 EXTRAITS DE CODE SOURCE ANDROID

Les extraits suivants présentent les composants clés de l'architecture MVVM du système Securia, implémentés en Kotlin avec Jetpack Compose.

A.2.1 HOMEVIEWMODEL ORCHESTRATEUR DE DÉTECTION

Le HomeViewModel est le composant central qui coordonne la lecture des capteurs, le calcul des caractéristiques SMA, et la décision de cascade IF + Autoencoder. Il expose un StateFlow d'état DetectionState observé par l'interface Compose.

```
@HiltViewModel
```

```
s HomeViewModel @Inject constructor( private val sensorRepository : SensorRepository,
   private val autoencoderModel     : AutoencoderAnomalyModel, private val
   notificationService: NotificationService, private val uploadEvidenceUseCase:
   UploadEvidenceUseCase
   ) : ViewModel() {

   private val _detectionState =
           MutableStateFlow<DetectionState>(DetectionState.Idle) val detectionState:
   StateFlow<DetectionState> = _detectionState

   /** Demarre la surveillance en continu via un Flow de batchs IMU
           */   fun    startSurveillance()    {    viewModelScope.launch    {
   sensorRepository.sensorBatchFlow()
ect { batch -> processBatch(batch)
   }
   }
   }

ate suspend fun processBatch(batch: SensorBatch) { // Etage 1 : Isolation Forest (score calcule cote
   serveur) val ifScore = batch.isolationForestScore() if (ifScore >= 0) {   // +1 Normal
   _detectionState.value = DetectionState.Normal return
   }

   // Etage 2 : Autoencoder Dense (inference TFLite embarquee)
```

```
        ıse = autoencoderModel.reconstructionError( floatArrayOf(batch.smaAcc, batch.smaGyr)
          )
          if (mse < autoencoderModel.threshold) {
          _detectionState.value = DetectionState.Suspect(mse) return
          }

          // Vol confirme par les deux etages
          _detectionState.value = DetectionState.TheftDetected(mse) handleTheftDetected(mse)
          }

        te suspend fun handleTheftDetected(score: Float) { notificationService.sendAlert(score)
          uploadEvidenceUseCase.captureAndUpload()
          }




}
```

Listing A.1 – HomeViewModel.kt – Logique principale d'orchestration

A.2.2 SENSORBATCHINGSERVICE ACQUISITION IMU PAR LOTS

Le SensorBatchingService est un ForegroundService Android qui enregistre les données des capteurs IMU (accéléromètre + gyroscope) à 20 Hz via SENSOR_DELAY_GAME et les expose sous forme de Flow de batchs de 10 échantillons.

```
@AndroidEntryPoint
class SensorBatchingService : Service(), SensorEventListener {
```

```
        @Inject lateinit var sensorManager: SensorManager

private val _batchFlow = MutableSharedFlow<SensorBatch>( extraBufferCapacity = 64
        )
        val batchFlow: SharedFlow<SensorBatch> = _batchFlow

        private val accBuffer = CircularBuffer(BATCH_SIZE) private val gyroBuffer =
        CircularBuffer(BATCH_SIZE)

companion object { const val BATCH_SIZE = 10 // echantillons par lot
        }

override fun onSensorChanged(event: SensorEvent) { when (event.sensor.type) {
        Sensor.TYPE_ACCELEROMETER -> accBuffer.add(event.values) Sensor.TYPE_GYROSCOPE ->
        gyroBuffer.add(event.values
        )
        }
if (accBuffer.isFull && gyroBuffer.isFull) { val batch = SensorBatch( accData =
        accBuffer.snapshot(), gyroData = gyroBuffer.snapshot(), smaAcc =
        computeSMA(accBuffer.snapshot()), smaGyr =
        computeSMA(gyroBuffer.snapshot()), timestamp = System.currentTimeMillis()
        )
        _batchFlow.tryEmit(batch) accBuffer.clear() gyroBuffer.clear()
```

```
        }
        }

    ate fun computeSMA(data: Array<FloatArray>): Float = data.map { v -> sqrt(v[0].pow(2)
        + v[1].pow(2) + v[2].pow(2))
        }.average().toFloat()

        override fun onStartCommand(intent: Intent?, flags: Int, startId
        : Int): Int {
    sorManager.registerListener( this, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER
        ),
        SensorManager.SENSOR_DELAY_GAME
        ) sensorManager.registerListener( this,
        sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
        SensorManager.SENSOR_DELAY_GAME
        ) return START_STICKY
        }



}
```

Listing A.2 – SensorBatchingService.kt – Lecture et batching des capteurs IMU

A.2.3 AUTOENCODERANOMALYMODEL INFÉRENCE TFLITE

La classe AutoencoderAnomalyModel encapsule le modèle TFLite et expose une méthode reconstructionError() utilisée par le HomeViewModel pour calculer l'erreur MSE en moins de 0,1 ms.

```
    class AutoencoderAnomalyModel @Inject constructor(
    @ApplicationContext private val context: Context
al threshold = 0.9681f // seuil theta P95 -- mode marche

ate val interpreter: Interpreter by lazy { val model = loadModelFile("autoencoder_anomaly_model.tflite"
    )
rpreter(model, Interpreter.Options().apply { setNumThreads(1)
    })
    }

* Retourne l'erreur de reconstruction MSE pour un vecteur [ smaAcc, smaGyr] */
reconstructionError(input: FloatArray): Float { val inputTensor = Array(1) { input } val
    outputTensor = Array(1) { FloatArray(input.size) } interpreter.run(inputTensor, outputTensor)
    return input.indices.map { i ->
    (input[i] - outputTensor[0][i]).pow(2)
    }.average().toFloat()
    }

    private fun loadModelFile(fileName: String): ByteBuffer {
```

```
                val fileDescriptor = context.assets.openFd(fileName) val inputStream =
                FileInputStream(fileDescriptor.
                fileDescriptor)
               rn inputStream.channel.map( FileChannel.MapMode.READ_ONLY,
                fileDescriptor.startOffset, fileDescriptor.declaredLength
                )



        }
}
```

Listing A.3 – AutoencoderAnomalyModel.kt – Inférence TFLite embarquée

A.3 SCRIPT D'ÉVALUATION PYTHON (GOOGLE COLAB)

Le script suivant a été utilisé pour évaluer les modèles ML de Securia à partir des notebooks Google Colab et des fichiers TFLite. Il est reproductible : toute personne disposant des datasets SecurIA_Dataset_marche.csv et SecurIA_Dataset_running.csv ainsi que des modèles .tflite peut obtenir les métriques présentées au Chapitre 4.

```
import numpy as np import
pandas as pd
from sklearn.ensemble import IsolationForest from sklearn.preprocessing import
StandardScaler from sklearn.metrics import f1_score, confusion_matrix import
tensorflow as tf


# --- Calcul SMA et chargement dataset --def load_dataset(path,
label_col='label'):
df = pd.read_csv(path) ax, ay, az = df['Acc_x'], df['Acc_y'], df['Acc_z']
```

```
df['SMA_acc'] = np.sqrt(ax**2 + ay**2 + az**2) gx, gy, gz = df['Gyro_x'], df['Gyro_y'], df['Gyro_z']
df['SMA_gyr'] = np.sqrt(gx**2 + gy**2 + gz**2) return df[['SMA_acc','SMA_gyr']].values,
df[label_col].values


# --- Evaluation Isolation Forest --def evaluate_isolation_forest(X_train, X_test,
y_test,
                                                contamination=0.10):
model = IsolationForest(contamination=contamination,
n_estimators=100, random_state=42)
model.fit(X_train) y_pred = np.where(model.predict(X_test) == -1, 1, 0) tn, fp, fn, vp =
confusion_matrix(y_test, y_pred).ravel() return model, y_pred, {
                                    'sensibilite': vp/(vp+fn), 'specificite': tn/(tn+fp),
'f1': f1_score(y_test, y_pred, zero_division=0)
}


# --- Evaluation Autoencoder TFLite --def evaluate_autoencoder_tflite(X_train, X_test,
y_test,
tflite_path, percentile=95):
interp = tf.lite.Interpreter(model_path=tflite_path) interp.allocate_tensors() in_det =
interp.get_input_details()[0] out_det = interp.get_output_details()[0]


run(X): results = [] for s in X:
interp.set_tensor(in_det['index'],
```

```
                                                    s.reshape(in_det['shape']).astype(np.
                                        float32))
interp.invoke() results.append(interp.get_tensor(out_det['index']).
flatten())
return np.array(results)

mse_train = np.mean((X_train - run(X_train))**2, axis=1) theta = np.percentile(mse_train,
percentile) mse_test = np.mean((X_test - run(X_test))**2, axis=1) y_pred = (mse_test >=
theta).astype(int) tn, fp, fn, vp = confusion_matrix(y_test, y_pred).ravel() return interp,
y_pred, {
'theta': theta, 'sensibilite': vp/(vp+fn),
'specificite': tn/(tn+fp),
'f1': f1_score(y_test, y_pred, zero_division=0)
}

# --- Cascade IF + AE ---
def evaluate_cascade(y_test, y_pred_if, y_pred_ae):
y_cascade = np.where((y_pred_if == 1) & (y_pred_ae == 1), 1, 0)
tn, fp, fn, vp = confusion_matrix(y_test, y_cascade).ravel() return y_cascade, {
'sensibilite': vp/(vp+fn), 'specificite': tn/(tn+fp),
'f1': f1_score(y_test, y_cascade, zero_division=0)
}
```

Listing A.4 – evaluation_securia.py – Fonctions principales d'évaluation (extrait)

Le script complet (incluant les fonctions de visualisation et de génération automatique des tableaux LATEX) est disponible dans le fichier evaluation_securia.py fourni en accompagnement de ce mémoire.