SIDE-CHANNEL ATTACKS ON IOT DATA AND COUNTERMEASURES

by
SEYEDPAYAM SEYEDKAZEMI

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Doctor of Philosophy

Sabancı University
July 2025

-

SIDE-CHANNEL ATTACKS ON IOT DATA AND COUNTERMEASURES

APPROVED BY

Prof. Dr. YÜCEL SAYGIN . . . .
(Dissertation Supervisor)

Prof. Dr. ALPTEKİN KÜPÇÜ . . . .

Assoc. Prof. Dr. ÖZNUR TAŞTAN . . . .

Assoc. Prof. Dr. BEKİR BEDİZ . . . .

Assoc. Prof. Dr. ALİ İNAN . . . .

DATE OF APPROVAL: July 23, 2025

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisors, Prof. Dr. Yücel Saygın and Asst. Prof. Dr. Mehmet Emre Gürsoy, for their unwavering guidance, support, and encouragement throughout my doctoral journey at Sabancı University. Their expertise in our field and their dedication to mentorship have played a pivotal role in shaping both my academic development and professional growth. In times of challenge, they consistently offered fresh perspectives, posed thought-provoking questions, and provided invaluable feedback that helped me strengthen and refine my work.

I am also sincerely thankful to Assoc. Prof. Dr. Öznur Taştan and Assoc. Prof. Dr. Bekir Bediz for their insightful input and support as members of my dissertation committee. In addition, I extend my appreciation in advance to Prof. Dr. Alptekin Küpçü and Assoc. Prof. Dr. Ali İnan, who will participate in my final dissertation defense and provide valuable feedback.

I gratefully acknowledge the support of The Scientific and Technological Research Council of Turkey (TÜBİTAK) through the 2244 Industrial PhD Program, under project number 119C045. Their funding made this work possible.

Finally, my heartfelt thanks go to my family, Mirjavad Seyedkazemi, Hamideh Farajzadeh, and Seyedpouya Seyedkazemi, for their sincere love, patience, and belief in me. Their constant encouragement and support have been the foundation of my perseverance, and without them, this dissertation would not have been possible.

ABSTRACT

SIDE-CHANNEL ATTACKS ON IOT DATA AND COUNTERMEASURES

SEYEDPAYAM SEYEDKAZEMI

MECHATRONICS ENGINEERING Ph.D DISSERTATION, JULY 2025

Dissertation Supervisor: Prof. Dr. YÜCEL SAYGIN
Dissertation Co-supervisor: Asst. Prof. Dr. MEHMET EMRE GÜRSOY

Keywords: Side-Channel Privacy Attacks, Ambient Light Sensor, Subject Inference, Sensor Data Privacy, Noise Injection Defense Mechanisms

This thesis explores novel privacy vulnerabilities arising from the unintended use of sensor data in mobile and wearable systems. The first part introduces LuxTrack, a side-channel attack that leverages the ambient light sensor (ALS) of a smartphone to infer user activities on nearby laptop screens based on emitted light intensity. We developed an Android application to collect ALS data in a controlled environment with real users and showed that machine learning models trained on extracted features could infer viewed websites or applications with up to 80% accuracy. We then proposed and evaluated three countermeasures—binning, smoothing, and noise addition—demonstrating that attack accuracy could be reduced to below 30% while maintaining legitimate task utility.

In the second part, we examine how motion sensor datasets, initially designed for activity or fall detection, can be exploited for subject inference attacks. Using the SisFall dataset, we show that it is possible to accurately identify individuals based on their motion patterns using machine learning models and statistically significant features. We propose and evaluate several defense mechanisms that inject noise at the feature and sensor levels, achieving a strong trade-off between reducing subject identification accuracy and preserving activity recognition performance.

Together, these studies highlight the risks of side-channel leaks and unintended inferences in sensor-based systems, and propose practical defenses to support privacy-preserving data analytics.

ÖZET

IOT VERILERI ÜZERINE YAN KANAL SALDIRILARI VE SAVUNMA MEKANIZMALARI

SEYEDPAYAM SEYEDKAZEMI

MEKATRONİK MÜHENDİSLİĞİ DOKTORA TEZİ, TEMMUZ 2025

Tez Danışmanı: Prof. Dr. YÜCEL SAYGIN
Tez Eş Danışmanı: Dr. Öğr. Üyesi MEHMET EMRE GÜRSOY

Anahtar Kelimeler: Yan Kanal Gizlilik Saldırıları, Ortam Işığı Sensörü, Kişi Çıkarımı, Sensör Verisi Gizliliği, Gürültü Enjeksiyonlu Savunma Mekanizmaları

Bu tez, mobil ve giyilebilir sistemlerdeki sensör verilerinin amaç dışı kullanımıyla ortaya çıkan yeni gizlilik açıklarını incelemektedir. İlk bölümde, bir akıllı telefonun ortam ışığı sensöründen (ALS) yararlanarak yakındaki bir dizüstü bilgisayar ekranında gerçekleştirilen kullanıcı aktivitelerini, ekrandan yayılan ışık yoğunluğuna göre tahmin eden bir yan kanal saldırısı olan LuxTrack tanıtılmaktadır. Gerçek kullanıcılarla kontrollü bir ortamda ALS verisi toplamak üzere geliştirdiğimiz Android uygulaması sayesinde, çıkarılan özelliklerle eğitilen makine öğrenimi modellerinin görüntülenen internet sitelerini veya uygulamaları %80'e varan doğrulukla tahmin edebildiğini gösterdik. Ardından, saldırı doğruluğunu %30'un altına indirirken, meşru görevlerin doğruluğunu yalnızca %3 oranında azaltan üç karşı önlem—aralıklama (binning), yumuşatma ve gürültü ekleme—önerdik ve değerlendirdik.

Tezin ikinci bölümünde, başlangıçta aktivite veya düşme tespiti amacıyla oluşturulan hareket sensörü veri kümelerinin, kişi çıkarımı saldırıları için nasıl kötüye kullanılabileceği ele alınmaktadır. SisFall veri kümesi kullanılarak, bireylerin hareket desenlerinden yola çıkarak, anlamlı istatistiksel özellikler ve makine öğrenimi modelleri yardımıyla yüksek doğrulukta tanımlanabildiği gösterilmiştir. Özellik ve sensör düzeyinde gürültü enjekte eden çeşitli savunma mekanizmaları önerilmiş ve bu mekanizmaların, aktivite tanıma performansını korurken kişi tanıma doğruluğunu önemli ölçüde düşürdüğü ortaya konmuştur.

Bu çalışmalar birlikte değerlendirildiğinde, sensör tabanlı sistemlerdeki yan kanal sızıntıları ve amaç dışı bilgi çıkarımı risklerine dikkat çekilmekte ve gizliliği koruyan veri analitiği için uygulanabilir savunma çözümleri sunulmaktadır.

To the roots of my strength and the light in my journey,
My family ...

TABLE OF CONTENTS

LIST OF TABLES .... **xi**

LIST OF FIGURES .... **xiii**

1. INTRODUCTION **1**

2. RELATED WORKS **6**

2.1. Side-Channel Attacks via Sensors: Focus on Activity Inference Exploiting Smartphone Ambient Light Sensors 6

2.1.1. Sensor-based Side-Channel Attacks on Smartphones 6

2.1.2. Ambient Light Sensors as a Source of Side-Channel Leakage 7

2.1.3. ALS-based Activity Inference Attacks 7

2.2. Datasets and Machine Learning Approaches for Fall Detection: A Review 8

2.2.1. Fall Detection Datasets: Characteristics and Benchmarking 8

2.2.2. Applications of Datasets and Machine Learning Methods in Fall Detection 11

3. ATTACKS METHODOLOGY: DESIGN AND IMPLEMENTATION **17**

3.1. LuxTrack: System Design and Implementation 17

3.1.1. Hardware and Software Setup 17

3.1.2. Data Collection 19

3.1.3. Feature Extraction and Selection 21

3.1.4. ML Model Training 26

3.2. Subject Inference Attack 27

3.2.1. Dataset 27

3.2.2. Feature Extraction and Selection 28

3.2.3. Subject Inference Attack Approaches 31

3.2.3.1. First cut solution 32

3.2.3.2. Proposed solution 35

4. ATTACKS RESULTS AND DISCUSSIONS 37
4.1. ALS-based Activity Inference Attack Results and Discussions 37
4.1.1. Comparison with LightSpy and Raw ALS Data 37
4.1.2. Analysis of Confusion Matrices 39
4.1.3. Feature Importance Analysis....................................... 40
4.1.4. Attack Robustness Evaluation 41
4.1.5. Results with LOO-CV 46
4.1.6. Intra-Activity Diversity 47
4.1.7. Extension to Multitasking.. 48
4.2. Subject Inference Attack Results and Discussions 50
4.2.1. Results of the first cut solution 50
4.2.2. Results of the Proposed Solution................................... 51

5. POTENTIAL COUNTERMEASURES AND THEIR IMPACT 56
5.1. Countermeasures against ALS-based Activity Inference Attack 56
5.1.1. Binning.. 57
5.1.2. Smoothing... 58
5.1.3. Noise Addition 59
5.1.4. Trade-offs Between Attack Accuracy and Legitimate Task Accuracy 60
5.1.5. Evaluation of Countermeasures on LightSpy 63
5.2. Countermeasures against Subject Inference Attack......................... 64
5.2.1. Features-Based Defense Mechanisms 66
5.2.1.1. Random Feature Noise Injection Defense Mechanism (RFNI) 66
5.2.1.2. Subject-Specific Feature Noise Injection Defense Mechanism (SFNI) 73
5.2.1.3. Subject-Only Feature Noise Injection Defense Mechanism (SOFNI) 75
5.2.1.4. Comparison of the Three Features-Based Defense Mechanisms 78
5.2.2. Raw-Sensor-Data-Based Defense Mechanisms...................... 79
5.2.2.1. Uniform Noise-Based Defense Mechanisms 79
5.2.2.1.1. Uniform Sensor-Axis Data Noise Injection Defense Mechanism (Uni-SANI) 80
5.2.2.1.2. Optimized Uniform Sensor-Axis Data Noise Injection Defense Mechanism (OptUni-SANI) 80
5.2.2.2. Sensor-Axis-Specific Noise Analysis 82

5.2.2.3. Optimized Sensor-Axis-Specific Noise Injection Defense Mechanisms ... 83

5.2.2.3.1. Minimized Noise Optimization-Based Defense Mechanism (MinNoise) ... 85

5.2.2.3.2. Accuracy-Constrained Optimization-Based Defense Mechanism (AcConstrain) ... 86

5.2.2.4. Comparison of the Three Raw-Sensor-Data-Based Defense Mechanisms ... 88

6. CONCLUSION AND FUTURE WORK ... 92

BIBLIOGRAPHY ... 94

LIST OF TABLES

3.1. Features extracted from the recorded ALS time series by LuxTrack. Parentheses next to the feature name indicate that multiple values (e.g., multiple coefficients) are extracted for that feature. Features that do not have parentheses contribute a single value (e.g., one coefficient). 22

4.1. Accuracy and F1 scores of our LuxTrack attack versus LightSpy versus using raw ALS time series recordings in classification (without feature extraction). Results show that LuxTrack outperforms its most relevant competitor, LightSpy, by roughly 20% in terms of accuracy and F1 score. .. 38

4.2. Analysis of feature importance in terms of feature categories. For each rank range (e.g., top 10, 11-20, 21-30), the table shows what percentage of features within that range fall under each feature category. For example, 50% of features in top 10 fall under SFC category, 10% of features in top 10 fall under CFC category, 60% of features in 11-20 fall under TIFC category, etc. 40

4.3. Accuracy and F1 scores of LuxTrack on an Asus laptop with IPS display. .. 42

4.4. Accuracy and F1 scores of LuxTrack under varying brightness levels. 42

4.5. Accuracy and F1 scores of LuxTrack under varying gamma settings. 43

4.6. Accuracy and F1 scores of LuxTrack under two alternative ambient light conditions: reduced light (the room is illuminated by a distant bedside lamp) and daylight (the room is illuminated by sunlight during day hours). 44

4.7. Accuracy and F1 scores of LuxTrack using two alternative smartphones (Nokia 3.1 Plus and Oppo A5s). 45

4.8. Accuracy and F1 scores of LuxTrack under two rotation settings: 30 vertical rotation away from the laptop screen, 45 horizontal rotation away from the laptop screen. 46

4.9. Comparison between LuxTrack, LightSpy and raw ALS when LOO-CV is used instead of 10-fold cross validation. 46

4.10. Accuracy and F1 scores of LuxTrack when each participant is asked to watch the same Youtube video. 48
4.11. Accuracy and F1 scores of LuxTrack when participants are multitasking with two activities. 50
4.12. Accuracy of subject prediction models in first cut solution 51
4.13. Partial accuracy of subject prediction models for trials of each activity type. While walking and jogging trials—regardless of speed—tend to yield lower subject prediction accuracy, certain complex seated or transition activities consistently result in higher accuracy across all four models. 52
4.14. Accuracy of subject prediction models for models selected to already predict the activity in our proposed solution. 53
4.15. Accuracy of activity prediction models in the 1st step of our proposed solution 53
4.16. Accuracy of subject prediction models on datasets of already predicted activities in the 1st step of our proposed solution 55

5.1. Impact of binning on legitimate task performance. 61
5.2. Impact of smoothing on legitimate task performance. 61
5.3. Impact of noise addition on legitimate task performance. 62
5.4. Summary of assumptions and information availability for each proposed defense mechanism. 67

LIST OF FIGURES

3.1. Hardware setup used in our data collection 17

3.2. Two sample screenshots from our Android application for recording ALS readings 19

3.3. Sample ALS values recorded from different activities performed by one participant 20

3.4. Subject Prediction Schematic in our First cut solution 34

3.5. Subject Prediction Schematic in our Proposed solution 35

4.1. Confusion matrices of GBT and NN models obtained after executing LuxTrack. 39

4.2. Confusion matrices of GBT and NN models in the simpler scenario where all users watch the same Youtube video. 49

5.1. Attack accuracy of three ML models with highest original accuracy (NN, GBT and K-NN) on binned data, with varying number of bins n. 57

5.2. Attack accuracy of three ML models (NN, GBT and K-NN) after smoothing, with a varying number of readings (data points) k used in smoothing. 58

5.3. Attack accuracy of three ML models (NN, GBT and K-NN) after noise addition, with varying standard deviation σ used in the generation of noise samples. 59

5.4. Trade-off between legitimate task accuracy (changing screen background color) and activity inference attack, for the binning countermeasure. 63

5.5. Trade-off between legitimate task accuracy (changing screen background color) and activity inference attack, for the smoothing countermeasure. 64

5.6. Trade-off between legitimate task accuracy (changing screen background color) and activity inference attack, for the noise addition countermeasure. 64

5.7. Attack accuracy of LightSpy models after our binning countermeasure is applied. 65

5.8. Attack accuracy of LightSpy models after our smoothing countermeasure is applied. 65
5.9. Attack accuracy of LightSpy models after our noise addition countermeasure is applied. 66
5.10. Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to all features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models. 69
5.11. Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to random 75% of features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models. 70
5.12. Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to random 50% of features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models. 71
5.13. Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to random 25% of features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models. 72
5.14. Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to subject-specific features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models. 74
5.15. Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to subject-only features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models. 77
5.16. Impact of noise intensity on subject prediction accuracy and activity prediction accuracy across the three defense mechanisms for LightGBM models. 79
5.17. Effect of Uni-SANI on the accuracy decline of (a) Subject prediction models and (b) Activity prediction models with increasing noise intensity, while (c) shows the comparative changes in accuracy between Subject and Activity prediction models. 81

5.18. Performance of the OptUni-SANI defense mechanism. (a) Depicts changes in Subject and Activity prediction accuracy for the LightGBM model at varying noise standard deviation (σ) values during the binary search process. (b) Illustrates the relationship between Subject prediction accuracy and Activity prediction accuracy, aiding in the selection of the optimal threshold value (α). 82

5.19. Results of Sensor-Axis-Specific Noise Analysis. (a) and (b) show how noise on the x- and y-axes of Acc2 impacts Activity but not Subject prediction accuracy. (c) Highlights accuracy reductions for both model types with noise on the y-axis of Acc1. (d), (e), and (f) depict the x- and z-axes of Acc1 and the x-axis of the Gyro sensor, where Subject prediction accuracy drops significantly while Activity prediction accuracy is preserved, with the Gyro x-axis showing the most desirable results. 84

5.20. (a) Minimum total noise intensities ($\sigma_1 + \sigma_2$) required to reduce Subject prediction accuracy below various thresholds (α) under MinNoise. The results highlight the trade-off between noise intensity and Subject prediction accuracy, with lower thresholds requiring higher combined noise intensities. Activity prediction accuracy remains unconstrained in this scenario. (b) The balance between Subject and Activity prediction accuracies under MinNoise, showing that a significant reduction in Subject prediction accuracy leads to only a marginal decline in Activity prediction accuracy. . . 87

5.21. Performance of Optimized Sensor-Axis-Specific Noise Injection under AcConstrain defense mechanisms. (a) Minimum achievable Subject prediction accuracy for different Activity prediction thresholds (α), demonstrating the trade-off between the two models. (b) Comparative accuracy reductions for Subject and Activity prediction models, showing minimal impact on Activity prediction despite significant degradation in Subject prediction accuracy. 89

5.22. Comparison of the three raw-sensor-data-based defense mechanisms in terms of Subject and Activity prediction accuracy using the LightGBM model. The figure includes: (1) results for Uni-SANI and OptUni-SANI with uniform noise application, (2) MinNoise with noise applied to the x-axis of the Gyro sensor, and (3) AcConstrain for both optimization conditions. The comparison illustrates how each mechanism balances the trade-off between reducing Subject prediction accuracy and preserving Activity prediction accuracy. 91

1. INTRODUCTION

Nowadays, mobile devices are equipped with a variety of sensors, including cameras, GPS, microphones, accelerometers, gyroscopes, and so forth. One of those sensors is the Ambient Light Sensor (ALS), which enables useful functionalities such as automatic adjustment of screen brightness and background color according to the ambient light of the environment. A smartphone app that uses the ALS is able to access and record the light intensity in the surrounding environment (in units of lux) without requiring explicit permission from the smartphone user Chakraborty, Ouyang & Srivastava (2017); Sikder, Petracca, Aksu, Jaeger & Uluagac (2021).

In the first part of this thesis, we demonstrate how an attacker can exploit the ALS of a smartphone, a seemingly innocuous sensor, to perform a side-channel attack to infer the user's activity on a nearby laptop using the light emitted from the laptop screen. Our attack, named *LuxTrack*, stems from the intuition that different activities, websites, and apps opened on the laptop's screen will have different background colors and different light fluctuations; therefore, their light emission characteristics will be different. For example, reading PDF documents will likely emit a constant amount of bright light, whereas watching YouTube videos will emit lower amounts or varying amounts of light due to darker backgrounds and dynamically changing colors in a video. Such differences in light emission characteristics can be continuously recorded by the smartphone's ALS and used to infer the user's activity.

The proliferation of sensor-based technologies has also revolutionized healthcare applications, particularly in activity recognition and fall detection for elderly populations. However, these same datasets may inadvertently expose sensitive information about individuals, raising critical privacy concerns. In the second part of this thesis, we investigate privacy vulnerabilities in motor signal datasets (e.g., accelerometer/gyroscope data) using the *SisFall* dataset as a case study. While such datasets are anonymized, we demonstrate that machine learning models can exploit unique biomechanical patterns to infer subjects' identities; a threat underexplored in prior work.

To demonstrate *LuxTrack*, we first developed an Android app that records the ambient light data from the ALS in units of lux. Using our app, we collected a new dataset from 10 human subjects in a controlled environment. Each subject was asked to perform 8 different everyday activities on the laptop, such as surfing on Facebook, chatting with friends on WhatsApp, reading PDF documents via Adobe Acrobat, and watching YouTube videos. To construct a sufficiently large dataset, each subject performed each activity 13 times (repetitions), and 120 ALS readings were collected in each repetition. We publicly release the Android app and the dataset we collected for the reproducibility of our results and to encourage further research in this field.

LuxTrack treats each unique recording in the dataset as a time series, and from each time series, it extracts a total of 187 features under 6 categories. Features used in *LuxTrack* are presented under categories based on their meaning and functionality, such as statistical features (e.g., mean, median, variance), change-based features (e.g., mean absolute change, absolute energy), transform-based features (e.g., Fast Fourier and Continuous Wavelet Transforms), and entropy-based features (e.g., binned entropy, permutation entropy, Lempel-Ziv complexity). *LuxTrack* then formulates the activity inference problem as an 8-class classification problem (each class corresponding to one activity type), and the extracted features are used to train machine learning (ML) models in a supervised fashion. Six ML models are trained, ranging from support vector machines (SVM) to neural networks (NN) and gradient boosted trees (GBT).

Similarly, in our analysis of the *SisFall* dataset, which records movements from 24 subjects performing 19 activities, we treat each trial as a time series and extract features to train ML models (XGBoost, LightGBM). Here, however, the threat model shifts from activity inference to *subject inference*, where an attacker identifies individuals from their kinematic signatures. Our experiments show this attack achieves up to 89.04% accuracy, highlighting the privacy risks even in anonymized datasets.

Experimental results for *LuxTrack* show that it can achieve up to 80% accuracy in inferring the user's activity on the laptop, whereas the accuracy of random guessing is 12.5%. Highest accuracy is achieved with NN, GBT, and K-NN (k-nearest neighbors) models, respectively. Also, we compare the results of *LuxTrack* against two benchmarks: (i) using raw ALS time series recordings in classification without feature extraction, and (ii) *LightSpy* Chakraborty et al. (2017), which is the most relevant work to ours from the literature. Results show that *LuxTrack*'s accuracy and F1 scores outperform raw ALS time series by approximately 25% and *LightSpy* by approximately 20%.

Motivated by the high accuracy of *LuxTrack*, we then explore countermeasures against it.

We propose three countermeasures which work by manipulating ALS readings: binning, smoothing, and noise addition. Binning discretizes ALS readings into n equal-sized bins, smoothing averages ALS readings using k nearest readings to them, and noise addition adds randomly and independently sampled Gaussian noise to each ALS reading with mean 0 and standard deviation σ. We experimentally show that these countermeasures are effective in reducing attack accuracy.

For motor signal datasets, we propose a taxonomy of lightweight countermeasures to mitigate subject inference risks. Unlike uniform noise injection, our defenses, such as *Subject-Only Feature Noise Injection* and *AcConstrain*, strategically perturb identity-revealing features or sensor axes (e.g., Gyro-x). These reduce subject inference accuracy to 14.47% with <5% utility loss for activity prediction, making them practical for edge deployment.

Although the proposed countermeasures for *LuxTrack* are effective in reducing attack accuracy, since they manipulate ALS readings, they also hurt the accuracy of legitimate tasks such as automatically adjusting screen background color. Therefore, a desirable goal is to achieve high protection from the *LuxTrack* attack with as little reduction in legitimate task accuracy as possible. Toward this goal, we perform a trade-off analysis between attack accuracy and legitimate task accuracy for all three countermeasures. Based on the results of our analysis, we identify and recommend appropriate trade-off settings and parameters for our countermeasures. For example, noise addition with $\sigma = 2$ is able to reduce attack accuracy to below 30% while maintaining legitimate task accuracy above 97%.

Similarly, our motor signal defenses optimize the privacy-utility trade-off by differentiating task-critical features (e.g., via p-value rankings) and employing Bayesian optimization for noise injection. This ensures minimal impact on activity recognition while maximizing privacy protection.

Contributions

This thesis makes the following contributions across its two parts:

- **Sensor-Based Side-Channel Attacks**:
  - We develop an Android app for collecting ALS readings and release a dataset of 10 subjects performing 8 laptop activities.
  - We propose *LuxTrack*, an ML-based attack achieving ∼20% higher accuracy than *LightSpy*, and evaluate it under diverse conditions (From different display

  - technologies and settings, smartphone models and ambient light conditions to extension toward intra-activity diversity and Multitasking).
  - We design three countermeasures (binning, smoothing, noise addition) with trade-off analysis for practical deployment.

- **Privacy Risks in Motor Signal Datasets**:
  - We demonstrate *subject inference attacks* on the *SisFall* dataset, achieving 89.04% accuracy using XGBoost/LightGBM.
  - We propose **feature-based** and **sensor-based defenses** (e.g., *AcConstrain*) that reduce subject inference to 14.47% with <5% utility loss.
  - We introduce a privacy-utility optimization framework for sensor data, addressing real-world constraints in wearables.

Organization

The rest of this thesis is organized as follows:

- **Chapter 2** reviews related work in two key areas: (i) sensor-based side-channel attacks with focus on Ambient Light Sensor (ALS) vulnerabilities (Section 2.1), and (ii) fall detection datasets and machine learning approaches that contextualize our motor signal privacy analysis (Section 2.2).
- **Chapter 3** presents the technical foundations of both attacks:
  - Section 3.1 details *LuxTrack*'s hardware/software setup (Section 3.1.1), data collection protocol (Section 3.1.2), and ML pipeline (feature extraction in Section 3.1.3, model training in Section 3.1.4).
  - Section 3.2 analyzes subject inference attacks on motor signal datasets, covering dataset characteristics (Section 3.2.1), feature engineering (Section 3.2.2), and attack methodologies (Sections 3.2.3.1–3.2.3.2).
- **Chapter 4** evaluates both attacks:
  - Section 4.1 compares *LuxTrack* against *LightSpy* (Section 4.1.1), analyzes confusion matrices (Section 4.1.2), investigates feature importance (Section 4.1.3), evaluates robustness (Section 4.1.4), and extends the analysis to LOO-CV (Section 4.1.5), Intra-Activity Diversity (Section 4.1.6), and multitasking scenarios (Section 4.1.7).

  - Section 4.2 presents subject inference accuracy for both first-cut (Section 4.2.1) and proposed solutions (Section 4.2.2).

- **Chapter 5** proposes defenses:

  - Section 5.1 evaluates binning, smoothing, and noise addition for ALS attacks (Sections 5.1.1–5.1.3), including trade-off analysis (Section 5.1.4).

  - Section 5.2 introduces feature-based (Section 5.2.1) and sensor-axis-specific (Section 5.2.2) countermeasures for motor signals.

- **Chapter 6** summarizes key findings, discusses limitations, and outlines future directions for sensor-based privacy protection.

2. RELATED WORKS

To contextualize our work, this chapter reviews relevant literature from two distinct domains that form the foundation of our research. The first part examines prior studies on sensor-based side-channel attacks, with a particular emphasis on activity inference attacks exploiting smartphone ambient light sensors. The second part focuses on datasets and machine learning approaches developed for fall detection systems, especially those that may inadvertently expose sensitive information such as subject identity. This dual-perspective review highlights the intersection of utility and vulnerability in sensor-driven systems, motivating the privacy-focused contributions of this thesis.

2.1 Side-Channel Attacks via Sensors: Focus on Activity Inference Exploiting Smartphone Ambient Light Sensors

Prior research on sensor-based side-channel attacks can be broadly grouped into three categories: general sensor-based attacks, threats specific to ambient light sensors, and ALS-based activity inference; the focus of our study.

2.1.1 Sensor-based Side-Channel Attacks on Smartphones

Smartphones contain numerous sensors such as temperature sensors, motion sensors, light sensors, etc., which can be used to execute side-channel attacks to infer users' sensitive information Delgado-Santos, Stragapede, Tolosana, Guest, Deravi & Vera-Rodriguez (2022); Sikder et al. (2021). In many works, accelerometer, gyroscope, and/or magnetometer sensors were used to perform keylogging Hussain, Al-Haiqi, Zaidan, Zaidan, Kiah, Anuar & Abdulnabi (2016); Javed, Beg, Asim, Baker & Al-Bayatti (2020); Marquardt, Verma, Carter & Traynor (2011) or infer the user's location Narain, Vo-Huu, Block & Noubir (2016); Nguyen, Akram, Markantonakis, Luo & Watkins (2019); Yang, Wu, Zhou, Zhang, Wang & Liu (2015). In Diamantaris, Marcantoni, Ioannidis & Polakis (2020), threats of mobile web browsing via HTML5 WebAPI were studied by monitoring popular websites' data access from the WebAPI JavaScript calls down to the Android system calls to reveal the extent to which websites are leveraging the WebAPI for collecting sensor data. A recent work by Mehrnezhad et al. Mehrnezhad, Makarouna & Gray (2022) performed a user study to evaluate user awareness, concerns, and preferences for mobile ambient sensors when accessed via apps and websites. Despite the risks, the authors conclude that the majority of the participants are not or only a little concerned about mobile ambient sensors. This may indicate the public's lack of awareness regarding the privacy and security risks of mobile sensors.

2.1.2 Ambient Light Sensors as a Source of Side-Channel Leakage

An early work which considered *light* as a source of side-channel attacks was proposed in Spreitzer (2014). This work argued that minor tilts and turns of smartphones cause variations in ALS readings on the smartphone. Through these variations, the authors demonstrated the possibility of inferring the user's PIN input from a set of known PINs. In another attack Mazilu, Blanke, Calatroni & Tröster (2013), low-power sensors such as pressure, temperature, humidity and light sensors were used to infer the user's semantic location (at home, in a shop, etc.). In Abe, Sato, Watanabe, Hashizume & Sugimoto (2021) and Sato, Shimada, Murakami, Watanabe, Hashizume & Sugimoto (2021), it was shown that indoor localization and navigation can be facilitated using light emissions and ALS. In Abe et al. (2021), 2D localization of a smartphone was studied, by receiving sinusoidally modulated light signals emitted by ceiling-mounted LEDs. An ALS mounted on the smartphone was used to measure the reflected illumination from the floor. In Sato et al. (2021), a visible-light positioning system that uses ALS in a LED-based room lighting was proposed. The light signal strength for each LED was used to localize the smartphone. Both works demonstrate the possibility of inferring users' locations via ALS. In Maiti & Jadliwala (2019), potential privacy and security risks of Internet-enabled smart lights are studied. The authors describe three specific attacks that can be carried out using smart lights, including the inference of users' audio and video playback through analysis of the light emitted by the bulbs, and the use of the bulbs' infrared capabilities to create a covert channel for exfiltrating users' private data.

2.1.3 ALS-based Activity Inference Attacks

More closely related to our work are Holmes, Desai & Nahapetian (2016); Sabra, Maiti & Jadliwala (2018); Schwittmann, Matkovic, Weis & others (2016) and Chakraborty et al. (2017). In Schwittmann et al. (2016), the ALS of a smartphone is used to infer the video playing on a TV screen. The proposed method can recognize the current TV channel and/or Youtube video using a recording length of 15 to 120 seconds, by measuring correlations (e.g., Pearson weighted correlation) between the TV light emissions and available reference patterns. Recording lengths used in this work are typically longer than ours, and only correlation with pre-recorded reference patterns is used for inference. In Holmes et al. (2016), a method is developed to distinguish keystrokes of a laptop keyboard using only the ALS readings from a smartwatch worn on the user's non-dominant hand. Authors in Holmes et al. (2016) also investigate the feasibility of capturing screen emanations for determining user browser usage patterns. In a similar work by Sabra et al. Sabra et al. (2018), keystrokes on an ATM keypad were inferred using the ALS of a smartwatch. The changing light intensities on and around an ATM keypad were used to infer keystrokes.

Most closely related to our work is Chakraborty et al. (2017), named LightSpy. In LightSpy, ALS on a smartphone was used to capture light samples emitted by the screen of a laptop. Eight features were extracted from these light samples, and they were used to train predictive ML models for activity inference. However, LightSpy's classification problem contains few classes (2 or 4). In contrast, our work contains 8 user activities, and therefore, we have an 8-class classification problem. Furthermore, as opposed to 8 features extracted by LightSpy, our feature extraction and selection process identifies 187 features which are relevant to classification. We perform an experimental comparison against LightSpy in Section **??** and show that our attack achieves approximately 20% higher accuracy and F1 score compared to LightSpy. Finally, LightSpy does not consider countermeasures or defenses, whereas we propose and experimentally analyze three countermeasures.

2.2 Datasets and Machine Learning Approaches for Fall Detection: A Review

Recent years have witnessed a growing interest in the development of fall detection systems, driven by the increasing need for reliable solutions to support aging populations. In this section, we review key contributions from the literature that explore the construction, evaluation, and application of datasets for fall detection, as well as the use of machine learning techniques for enhancing detection accuracy. The review is organized into two main parts: the first focuses on prominent datasets and their characteristics, while the second examines how these datasets have been utilized in various fall detection frameworks, highlighting current methodologies, challenges, and advancements.

2.2.1 Fall Detection Datasets: Characteristics and Benchmarking

A creditable dataset produced for the purpose of fall detection study is t-fall by Medrano, Igual, Plaza & Castro (2014). This study addresses the challenge of efficiently detecting falls in the elderly population, an ongoing public health concern. While many studies rely on acceleration data from smartphones and employ supervised machine learning algorithms for fall detection, these approaches often lack representation of real-world elderly falls and may not generalize well to new situations. To address this, the authors propose using novelty detection methods that identify abnormal movements relative to activities of daily living (ADL), allowing for adaptability to new scenarios through continuous retraining. The study compares a nearest neighbor-based novelty detection approach with a traditional supervised Support Vector Machine (SVM) method using datasets collected from smartphone recordings of simulated falls and real-life ADLs. Despite the absence of elderly data, the results suggest that while the nearest neighbor method shows promise for adaptability, in most cases, a generic SVM performs better.

In another work (Vilarinho, Farshchian, Bajer, Dahl, Egge, Hegdal & Weggersen, 2015), authors collect similar data to investigates the potential of utilizing off-the-shelf consumer mobile devices, specifically a smartwatch and smartphone, as ubiquitous automatic fall detectors for seniors. Combining threshold-based and pattern recognition techniques, the implemented system achieved a 63% accuracy in detecting falls and 78% accuracy in identifying activities of daily living. While the smartwatch sensors and algorithms marginally contributed to the system's accuracy, adjustments to thresholds and fuzzification are suggested to further enhance performance. Additionally, the open-source nature of the work aims to facilitate threshold tuning and provide a benchmark for researchers in this field.

The authors (Frank, Vera Nadales, Robertson & Pfeifer, 2010) introduce an activity recognition system focused on seven crucial motion-related activities, utilizing only an Inertial Measurement Unit (IMU) worn on the belt. Bayesian techniques, informed by real-time calculations of relevant IMU raw data features, are applied for classification, with a K2 learning algorithm constructing the Bayesian Network (BN) based on supervised data. Comparative analysis involving dynamic and static inference algorithms, evaluated on labeled datasets from 16 subjects, reveals that a Hidden Markov Model (HMM) derived from the learned BN yields the most effective results.

Fall detection, crucial in preventive medicine and assisted living, particularly for the elderly, has spurred the collectors of MobiFall (Vavoulas, Pediaditis, Spanakis & Tsiknakis, 2013) to employ accelerometers and gyroscopes to develop a system and latter offer additional benefits such as immediate emergency communication. They introduce a human activity dataset aimed at facilitating the testing and comparison of fall detection and activity recognition algorithms using smartphone inertial sensor data. The dataset includes signals from accelerometer and gyroscope sensors for 4 types of falls and 9 activities of daily living. Initial evaluations of 3 fall detection algorithms using this dataset are presented. In the updated version of their work (Chatzaki, Pediaditis, Vavoulas & Tsiknakis, 2017; Vavoulas, Chatzaki, Malliotakis, Pediaditis & Tsiknakis, 2016), the authors present a comprehensive evaluation of utilizing smartphone acceleration sensors for both human activity and fall recognition. Recorded from 66 subjects, the "MobiAct" dataset comprises 12 activities of daily living (ADLs) and 4 types of falls, serving as a benchmark for developing recognition systems. The study proposes optimized feature selection and classification schemes, achieving high accuracies of 99.9% for recognizing common ADLs and 96.8% for the more complex task of recognizing all 12 ADLs and 4 falls.

Martínez-Villaseñor, Ponce, Brieva, Moya-Albor, Núñez-Martínez & Peñafort-Asturiano (2019) present UP-Fall Detection Dataset. The dataset comprises raw and feature sets retrieved from 17 healthy young individuals without any impairment that performed 11 activities and falls, with three attempts each. The dataset also summarizes more than 850 GB of information from wearable sensors, ambient sensors and vision devices. The aim of this dataset is mentioned to be help human activity recognition and machine learning research communities to fairly compare their fall detection solutions. It also provides many experimental possibilities for the signal recognition, vision, and machine learning community.

Addressing the significant demand for low-cost fall detection systems in an aging society, authors (Kwolek & Kepski, 2014) present a solution to the issue of false alarms commonly associated with inertial sensor-based detectors. For this purpose, they introduce the UR Fall Detection Dataset (URFD), comprising images of normal activities alongside instances of a person lying on the floor, and additional image sequences recorded in typical room settings. The dataset includes a total of 612 images, with 402 depicting typical activities of daily living (ADLs) and 210 showing individuals lying on the floor. Features extracted from depth images are utilized for analysis. The UR Fall Detection dataset encompasses 30 image sequences featuring 30 falls, performed by 5 individuals from standing and sitting positions, with RGB and depth images synchronized with motion data from x-IMU inertial devices.

The authors introduce KFall, a large-scale motion dataset derived from 32 participants performing 21 activities of daily living and 15 simulated falls while wearing an inertial sensor (Yu, Jang & Xiong, 2021). With synchronized motion videos providing temporal labels, KFall becomes the first dataset suitable for pre-impact fall detection. They also present 3 algorithm types (threshold-based, support vector machine, and deep learning) tailored to the KFall dataset. Deep learning achieves high overall accuracy (99.32%) and balanced sensitivity (99.01%) and specificity (99.77% and 94.87% for support vector machine). Their findings are claimed to establish a benchmark for future algorithm development and proactive injury prevention strategies for the elderly.

Another article (Saleh, Abbas & Le Jeannès, 2021) addresses the shortcomings of current fall detection datasets, including sensor types, placement, sampling frequency, and simulation protocols. To mitigate these issues, a comprehensive data acquisition system and simulation protocol are proposed, resulting in the creation of the FallAllD dataset. Comprising 26,420 files collected from sensors worn at the waist, wrist, and neck, FallAllD features motion signals captured by accelerometers, gyroscopes, magnetometers, and barometers. Deep learning and classical learning-based algorithms are evaluated on this dataset, revealing significant performance variations compared to reference datasets. Moreover, an in-depth analysis of acceleration-based fall detection identifies key factors contributing to false positives and false negatives.

In this paper, we utilized the *Sisfall* dataset (Sucerquia, López & Vargas-Bonilla, 2017) which contains the most extensive records from diverse subjects and activity types in contrast to other datasets. The authors in the related paper introduces a novel dataset comprising falls and activities of daily living (ADLs), collected using a custom wearable device equipped with accelerometers and a gyroscope. The dataset includes diverse ADLs and fall types performed by both young adults and elderly participants. Utilizing common feature extraction techniques and threshold-based classification, the dataset achieves high fall detection accuracy of up to 96%. However, further analysis reveals areas for improvement, particularly in certain activities where errors are concentrated, highlighting the need for novel approaches.

2.2.2 Applications of Datasets and Machine Learning Methods in Fall Detection

Falls pose a significant health risk for older individuals, necessitating urgent development of detection and prevention systems, particularly given the increasing aging population.

Authors (Pannurat, Thiemjarus & Nantajeewarawat, 2014) review existing fall detection systems and research challenges within the field, categorizing platforms into wearable and ambient devices, and classification methods into rule-based and machine learning techniques. Their analysis highlights merits, drawbacks, and outstanding research challenges for emerging platforms aiming to enhance fall detection and prevention for elderly individuals living independently or in care facilities.

In another paper by Casilari, Luque & Morón (2015), researchers conduct a comprehensive analysis of existing fall detection systems based on Android devices, systematically categorizing and comparing proposals based on system architecture, employed sensors, detection algorithms, and response mechanisms to fall alarms. Their review highlights the lack of a standardized framework for validation and comparison, as well as the need for evaluating the practical feasibility of utilizing Android devices with limited resources for fall detection solutions.

While past literature has primarily focused on fall detection using statistical and threshold-based approaches, the authors shift attention to the latest trends in fall detection and prevention systems leveraging Machine Learning (ML) algorithms (Usmani, Saboor, Haris, Khan & Park, 2021). Through analysis of recent studies, datasets, age groups, ML algorithms, sensors, and locations, they provide insights into current trends and future directions, aiming to aid researchers in addressing existing challenges and proposing innovative methodologies.

The authors in a related study (Mauldin, Canby, Metsis, Ngu & Rivera, 2018) introduce SmartFall, an Android app utilizing accelerometer data from a smartwatch IoT device to detect falls in real-time, preserving data privacy by performing computations locally on a paired smartphone. Experimentation with traditional and non-traditional machine learning algorithms, including Deep Learning, using three fall datasets (Smartwatch, Notch, Farseeing), reveals that Deep Learning models generally outperform traditional approaches. This superiority is attributed to Deep Learning's capacity to automatically learn subtle features from raw accelerometer data, enhancing generalizability to new users, a crucial aspect for real-world applicability. Additionally, they present a three-layer open IoT system architecture which facilitates the collection and analysis of various sensor data modalities for remote monitoring of a subject's wellbeing.

A novel pipeline for fall detection based on wearable accelerometer data, utilizing feature reduction techniques and classical machine learning algorithms is proposed (Al Nahian, Ghosh, Al Banna, Aseeri, Uddin, Ahmed, Mahmud & Kaiser, 2021). Validation using 3 publicly available datasets demonstrates the pipeline's superior efficiency in detecting falls, outperforming existing methods across all datasets and showcasing its generalization capability. The proposed data analysis pipeline highlights promising advancements in the field of elderly fall detection and prediction.

The authors (Dhiman & Vishwakarma, 2019) provide a comprehensive overview of existing approaches to abnormal human activity recognition (AbHAR), encompassing both handcrafted and deep learning methods. It addresses the diverse aspects influencing AbHAR systems, including anomaly definition, feature representation, application contexts, and datasets. Various feature designs for AbHAR in videos are discussed, considering applications such as fall detection, Ambient Assistive Living (AAL), homeland security, surveillance, and crowd analysis using RGB, depth, and skeletal evidence. The paper outlines key contributions and limitations of feature design techniques within the contexts of 2D and 3D AbHAR, offering insights into different approaches for abnormal action detection. Additionally, newly introduced datasets for AbHAR are presented, providing researchers with more complex validation methods.

The critical need for remote monitoring and early detection of falls among older adults in telemedicine is addressed by Mrozek, Koczur & Małysiak-Mrozek (2020). The writers present a scalable architecture capable of monitoring thousands of individuals, detecting falls, and notifying caregivers. The study includes scalability tests to determine operational requirements for large-scale systems and evaluates various Machine Learning models for fall detection, with Boosted Decision Trees exhibiting the best performance. Additionally, experiments comparing fall detection in both Cloud-based data centers and Edge IoT devices reveal significant reductions in data transmission size when processing falls at the Edge.

Bet, Castro & Ponti (2019) discover a scarcity of review papers covering the three main applications related to falls: detection, classification, and risk screening. They believe, this systematic review aims to identify the current state of fall event detection in older individuals using wearable sensors, along with key study characteristics and gaps in the literature. Out of 608 studies, 29 were included, revealing accelerometers as the most common sensor type, primarily placed at the waist or lumbar region with sampling rates of 50 or 100Hz. Methods comparing features extracted from accelerometer signals predominated, with fall risk screening being the most observed application. While this review offers valuable insights for future research, certain aspects such as sample size and data acquisition methods still lack consensus.

A wearable sensor-based continuous fall monitoring system is proposed by Hussain, Hussain, Ehatisham-ul Haq & Azam (2019), capable of not only detecting falls but also identifying falling patterns and associated activities. The system's performance is evaluated using three machine learning algorithms: k-nearest neighbors (KNN), support vector machine (SVM), and random forest (RF). Results show that the proposed methodology achieves high accuracy, with KNN classifier reaching 99.80% accuracy for fall detection and RF classifier achieving 96.82% accuracy for recognizing different falling activities.

A computational system capable of efficiently detecting and classifying falls is developed for elderly population monitoring and rapid assistance, mitigating the risk of prolonged injuries and fatalities by Galvão, Ferreira, Albuquerque, Barros & Fernandes (2021).

Addressing the challenge of false positives, authors propose various topologies of a multimodal convolutional neural network trained on RGB images and accelerometer data to detect falls. Evaluation on the UR-Fall Detection and UP-Fall datasets, alongside a comparison with state-of-the-art models, demonstrates promising results. Their model achieves state-of-the-art performance on the UP-Fall dataset and exhibits scalability and robustness for real-world fall detection using readily available sensors.

A machine learning framework for fall detection and daily activity recognition is presented (Chelli & Pätzold, 2019), utilizing acceleration and angular velocity data from two public databases. Time- and frequency-domain features are extracted from the data and provided to four classification algorithms: artificial neural network (ANN), K-nearest neighbors (KNN), quadratic support vector machine (QSVM), and ensemble bagged tree (EBT). Results demonstrate high accuracy, with the QSVM and EBT algorithms achieving 97.2% and 99.1% accuracy in fall detection without false alarms. Furthermore, additional features extracted from autocorrelation and power spectral density data improve classification accuracy, with the QSVM and EBT algorithms achieving 100% accuracy in fall detection without false alarms, representing the best achievable performance.

The authors (Yacchirema, De Puga, Palau & Esteve, 2018) propose an innovative IoT-based fall detection system for indoor environments, leveraging low-power wireless sensor networks, smart devices, big data, and cloud computing. The system utilizes a 3D-axis accelerometer embedded in a 6LowPAN wearable device to collect real-time movement data from elderly individuals. Sensor readings are processed using a decision trees-based Big Data model on a Smart IoT Gateway for efficient fall detection. Upon detection, alerts are triggered, and notifications are sent to caregivers. Additionally, the system offers cloud-based services, including storage for healthcare professionals to access fall data and the creation of machine learning models based on detected falls. Experimental results demonstrate high success rates in fall detection accuracy, precision, and gain.

In another work, the authors claim that while fall detection has been extensively studied, designing accurate yet computationally efficient algorithms for wearable devices remains a challenge, thus they propose a low-cost, highly accurate machine learning-based fall detection algorithm (Saleh & Jeannès, 2019). It introduces a novel online feature extraction method leveraging the time characteristics of falls and a machine learning system design optimized for accuracy and computational efficiency. Experimental results on a large dataset demonstrate over 99.9% accuracy with computational costs below 500 floating point operations per second, facilitating embedding in wearable sensors with minimal power requirements and enhanced autonomy.

A convolutional deep neural network's performance in identifying fall patterns using data from a transportable tri-axial accelerometer is evaluated by Casilari, Lora-Rivera & García-Lagos (2020). Unlike many previous studies, the evaluation encompasses a wide range of public data repositories containing traces from diverse volunteer groups performing both Activities of Daily Life (ADLs) and simulated falls. While the method shows promising results when hyper-parameterized for specific datasets, challenges arise in generalizing the network architecture across different testbeds, emphasizing the need for adaptable algorithms in real-world applications.

Researchers (Luna-Perejón, Domínguez-Morales & Civit-Balcells, 2019) investigate the feasibility of using recurrent neural network (RNN)-based models to detect falls and fall risks in real-time using accelerometer signals. Four different architectures are evaluated using *SisFall* dataset at varying frequencies, with resulting models integrated into two embedded systems to assess execution times and effectiveness. The simplest models achieved inference times below 34 ms, demonstrating the capability to detect fall events in real-time with high energy efficiency. Their findings suggest that RNN models offer an effective method for implementing autonomous wearable fall detection systems on low-power microcontrollers.

A software architecture based on RNN for fall detection is presented by Musci, De Martini, Blago, Facchinetti & Piastra (2020), designed to run entirely on wearable devices. Leveraging *SisFall* dataset (Sucerquia et al., 2017) with fine-grained temporal annotations, authors demonstrate that careful architectural minimization and hyperparameter selection yield a competitive model. Validation on state-of-the-art hardware confirms the feasibility of onboard implementation for effective fall detection.

In another paper, Sarabia-Jácome, Usach, Palau & Esteve (2020) address the critical issue of fall accidents in the elderly population, emphasizing the need for prompt assistance to mitigate health risks. Leveraging Ambient Assisted Living (AAL) technologies such as IoT, Cloud Computing, and Machine Learning (ML), the study integrates Deep Learning (DL) techniques for enhanced fall detection accuracy. By utilizing fog computing for ML inference, the system overcomes cloud-related latency issues, presenting an efficient fog-cloud architecture for real-time fall detection. Through the deployment of DL models on resource-constrained fog nodes, the system achieves high accuracy (98.75%), reduced delay, and improved service delivery compared to conventional fall detection systems.

Yu, Qiu & Xiong (2020) aim to develop deep neural networks to predict falls in older adults during their initiation and descent before impact, allowing for timely safety interventions. Three-class classification models were created using convolutional neural networks (CNN), long short-term memory (LSTM), and a hybrid model (ConvLSTM) using inertial sensor data. The ConvLSTM model exhibited superior sensitivity (93.15% non-fall, 93.78% pre-impact fall, 96.00% fall) and specificity (96.59%, 94.49%, 98.69%) compared to LSTM and CNN models, with minimal latency (1.06 ms) suitable for real-time implementation. These results suggest the hybrid ConvLSTM model's potential for embedded fall prediction systems to prevent fall-related injuries in older adults effectively.

Writers (Wu, Zheng, Chu, Cheng & Kim, 2022) introduce a novel Deep Learning (DL) model, Gated Recurrent Units (GRU), for fall detection, aiming to improve accuracy and efficiency. Compared with conventional Machine Learning (ML) methods, the GRU architecture demonstrated superior performance across multiple metrics on two widely-used datasets, MobiAct (Vavoulas et al., 2016) and Smartwatch (Mauldin et al., 2018), collected from mobile sensors. With a prediction accuracy of 99.56% and 90.69%, and F1 score of 96.83% and 87.29%, respectively, the proposed method shows promising potential for robust fall detection.

Another study by Delgado-Escaño, Castro, Cózar, Marín-Jiménez, Guil & Casilari (2020) introduces a novel deep learning-based approach for fall detection and individual identification, adaptable across diverse datasets without requiring parameter adjustments. Leveraging raw inertial data through multi-task learning, the proposed model achieves over 98% accuracy in fall detection and demonstrates robust performance in identifying individuals, with minimal false positives. Remarkably, this single-model solution enables real-time execution, showcasing its versatility across varying conditions and obviating the need for retraining, thereby facilitating practical deployment in real-world scenarios.

Authors investigate the efficacy of metaheuristic (MH) optimization algorithms in enhancing human activity recognition (HAR) and fall detection based on sensor data (Al-qaness, Helmi, Dahou & Elaziz, 2022). Nine MH algorithms are employed for feature selection to improve classification accuracy, including Aquila optimizer, arithmetic optimization algorithm, marine predators algorithm, artificial bee colony algorithm, genetic algorithm, slime mold algorithm, grey wolf optimizer, whale optimization algorithm, and particle swarm optimization algorithm. Preprocessing and segmentation methods are applied to reveal motion patterns and reduce time complexities, while a light feature extraction technique employing ResRNN model integrates convolution neural networks, residual networks, and bidirectional recurrent neural networks. Support vector machine and random forest classifiers are utilized for multi-classification and binary classification tasks across seven complex datasets, demonstrating promising performance of MH optimization algorithms in HAR and fall detection applications.

3. ATTACKS METHODOLOGY: DESIGN AND IMPLEMENTATION

3.1 LuxTrack: System Design and Implementation

The development and execution of LuxTrack consists of three main steps. First, we implemented an Android app for recording ALS readings (in units of lux) and collected ALS data from human volunteers using our Android app. Each recording from each volunteer was stored as a time series. Second, we extracted 187 features belonging to 6 categories from each time series. Finally, we trained 6 different machine learning models using the extracted features via supervised learning to infer the user's laptop activity. In this section, we describe these steps in more detail.

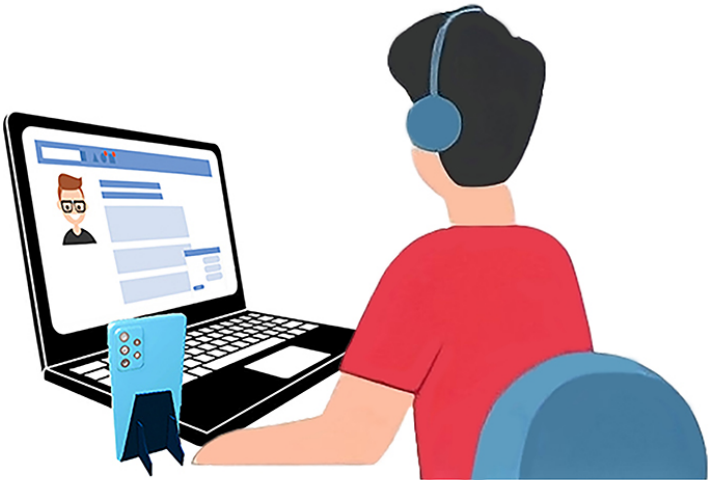


Figure 3.1 Hardware setup used in our data collection

3.1.1 Hardware and Software Setup

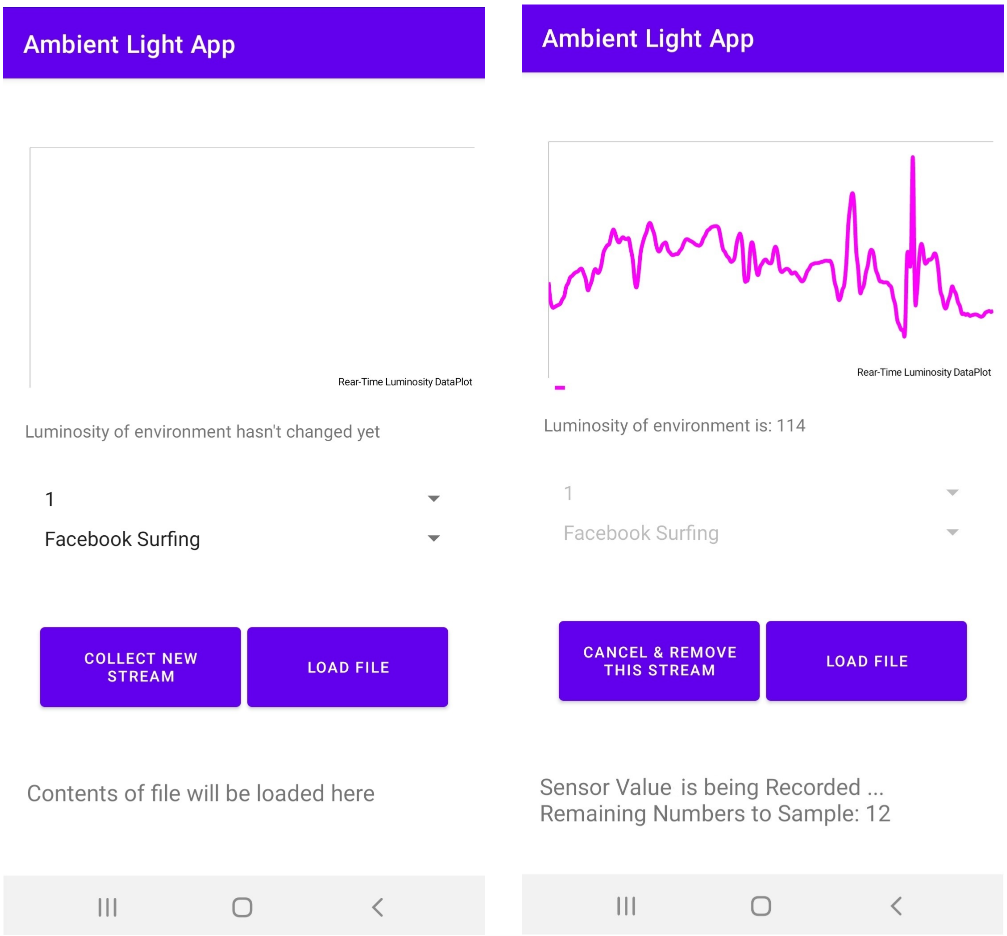
A visual representation of the hardware setup used in our data collection is shown in Figure 3.1. We used the ALS in a Samsung Galaxy A72 smartphone to record the light emitted from a flat-panel laptop screen through an Android application we developed and ran on the smartphone. The laptop used in our hardware setup was an E480 model

Lenovo laptop[1] with an Intel(R) UHD Graphics 620 display adapter, TN display, and 1366×768 screen resolution. The laptop screen was set to 100% brightness, and the screen lid was opened 100 degrees. The smartphone was placed near the keyboard of the laptop, its ALS pointing towards the laptop screen. Users were asked to perform various daily activities on the laptop, such as online shopping, watching videos, programming, etc. While the users were performing these activities, light emissions from the laptop's screen were recorded using our app on the smartphone. All data collection was carried out at night time in the same room illuminated by a light bulb, to ensure that ALS readings are not impacted by the existence of sunlight and/or lighting of the room.

We developed a custom Android app to record and store ALS readings on the Samsung Galaxy A72 smartphone. Two screenshots from our Android app are shown in Figure 3.2. The screenshot on the left shows the state of the app before data recording begins. Here, the researcher supervising the data collection enters or selects the pseudonymized user ID for the current laptop user. (Each laptop user participating in our data collection was assigned a random user ID to preserve their anonymity.) The supervisor also selects the activity being performed by the laptop user, e.g., Facebook surfing. After the supervisor presses the "Collect New Stream" button on the Android app, the app enters a waiting period of 5 seconds and displays a message to inform the supervisor. This 5-second period enables the placement of the smartphone near the laptop's keyboard and ensures that the laptop user begins the selected activity.

In Figure 3.2, the screenshot on the right shows the state of the app after data recording begins. The app records 4 samples per second (250 ms between each sample) from the smartphone's ALS. Although our app ran in the foreground of the smartphone during data collection, we also performed tests running the app in the background. We observed that there are no differences regarding the sampling rate or the permission requirements of the app, regardless of whether it is running in the background or foreground. The unit of illuminance recorded from the ALS is lux. For our app to be able to access the ALS, we import the SENSOR, SENSOREVENT, SENSOREVENTLISTENER, and SENSORMANAGER classes from the ANDROID.HARDWARE namespace. This namespace provides support for hardware features and sensors on Android. Our app does not require explicit user permissions to access and record ALS readings, which implies that any other app running on an Android device would be able to access and record ALS readings using our method without requiring extra permissions.

[1] Full product specifications: `https://psref.lenovo.com/syspool/Sys/PDF/ThinkPad/ThinkPad_E480/ThinkPad_E480_Spec.PDF`



Ambient Light App

Luminosity of environment hasn't changed yet

1

Facebook Surfing

Contents of file will be loaded here

Ambient Light App

Luminosity of environment is: 114

1

Facebook Surfing

Sensor Value is being Recorded ...
Remaining Numbers to Sample: 12

Rear-Time Luminosity DataPlot
COLLECT NEW STREAM
LOAD FILE
Rear-Time Luminosity DataPlot
CANCEL & REMOVE THIS STREAM
LOAD FILE

Figure 3.2 Two sample screenshots from our Android application for recording ALS readings

It can be observed from the screenshot on the right of Figure 3.2 that it is not possible to change the user ID or activity type during the recording phase. The recorded ALS readings are shown visually on the screen (topmost part) as a continuous time series. Each data collection lasts 30 seconds, causing the length of each time series to be 30 * 4 = 120 samples. During or after data collection, if it is found that the user ID or activity type was selected incorrectly, the current time series can be deleted using the "Cancel and Remove Stream" button on the app. Otherwise, the app is instructed to save the recorded time series to a local file.

3.1.2 Data Collection

Our data collection was approved by the Sabanci University Research Ethics Council (SUREC) under protocol number FENS-2023-69. All participants were informed about the procedure and signed a written consent form. A group of 10 participants voluntarily participated in the data collection. The participants consisted of 7 males and 3 females,

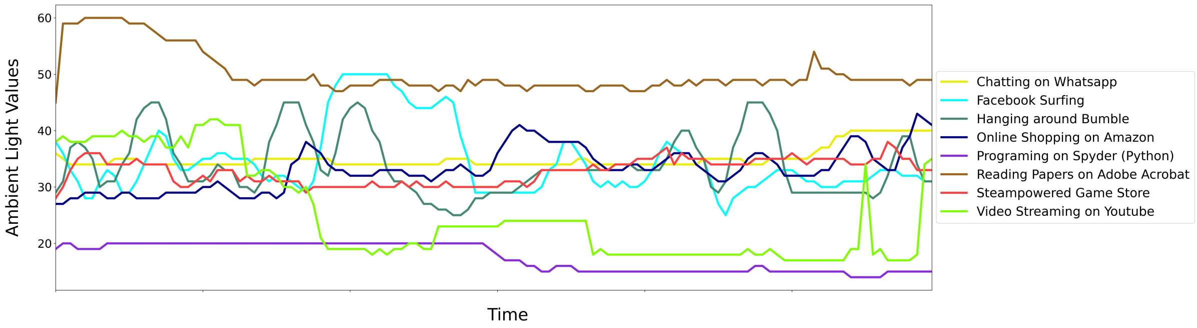


Figure 3.3 Sample ALS values recorded from different activities performed by one participant

aged between 24-37. Each participant was asked to perform the following 8 activities:

- Surfing on Facebook
- Surfing on Steampowered game store
- Using Bumble (online dating app)
- Chatting with friends on WhatsApp
- Programming on Spyder (Python IDE)
- Reading pdf documents via Adobe Acrobat
- Online shopping via Amazon
- Watching Youtube videos

These activities were selected because they are common everyday activities for many laptop users. In addition, many of the aforementioned websites are part of Alexa Top 100 most visited websites. Each participant was asked to perform each activity 13 times, resulting in 13 * 8 = 104 time series (each time series lasts 30 seconds) per participant. We note that participants were free to choose which pdf document to read, Youtube video to watch, or Amazon page to visit, etc. Participants logged into their own accounts on Facebook, Steam, WhatsApp, and similarly, they brought their own pdf document to read and their own Python code to work on.

Data collection was performed in the evening hours after sunset in a bedroom with a ceiling lamp. The ceiling lamp was turned on for the whole duration of the data collection. Collecting data in this setup eliminated the effect of changing daylight conditions. For example, the ambient light intensity can differ in the morning when compared to the noon, as well as on a sunny versus a cloudy day. We remain unaffected by such factors by collecting our data in the evening and in a room illuminated by the same ceiling lamp.

In Figure 3.3, we plot the ALS values recorded from one participant performing the 8 aforementioned activities. It can be observed that the ALS recordings corresponding to each different activity have different characteristics. For example, when the user is reading pdf files on Adobe Acrobat, the ALS values are consistent and high because pdf files typically have a white background, and the background does not change. Thus, the light emitted by the laptop screen is bright and remains consistent. In contrast, since Spyder has a dark background color, when the user is programming on Spyder, the ALS values are consistently low. We see fluctuating ALS values when the user is surfing on Facebook or watching Youtube videos, since the color of the image on the screen can vary greatly depending on the current image/video frame the user is viewing. LuxTrack exploits these different ALS characteristics of different activities in a methodical way to infer user activity.

Finally, we publicly release the Android app and the dataset we collected for the benefit of the research community and to encourage further advancements in this field.[2]

3.1.3 Feature Extraction and Selection

LuxTrack performs feature extraction and selection on the raw ALS time series data to obtain features that are relevant to activity inference. We initially extracted a total of 618 features, but 217 features were found to be significant (with p-value ≤ 0.05). These 217 features were further filtered down to 187 most significant features by decreasing the False Discovery Rate (FDR) from 0.05 to 5e-12. This feature selection was performed to eliminate redundant features as well as to reduce the computational cost of training downstream ML models used in the attack.

The full list of 187 features used in LuxTrack is given in Table 3.1. Based on their meaning and functionality, we present the features under 6 categories: statistical features category (SFC), change-based features category (CFC), value-based frequency features category (VFFC), trend-identifying features category (TIFC), transform-based features category (TFC), and entropy-based features category (EFC). Some features extract multiple values from each ALS time series sample, e.g., Fast Fourier Transform (FFT) has 21 coefficients. Such cases are denoted by parentheses, e.g., *FFT Coefficients (21).* If there are no parentheses, then the feature contributes one value. Below, we explain the features in each category one by one.

[2] Links to our Android app and dataset will be placed here upon paper acceptance.

Table 3.1 Features extracted from the recorded ALS time series by LuxTrack. Parentheses next to the feature name indicate that multiple values (e.g., multiple coefficients) are extracted for that feature. Features that do not have parentheses contribute a single value (e.g., one coefficient).

Feature Category	**Feature Name**
Statistical Features Category (SFC)	Minimum, Maximum, Mean, Median, Quantile (8), Variance, Standard Deviation, Range, Skewness, Variation Coefficient, Large Standard Deviation (3), Variance Larger Than Standard Deviation, Symmetry (5)
Change-based Features Category (CFC)	Mean Absolute Change, Absolute Sum of Changes, CID Complexity, Change Quantiles (42), Sum of Values, Absolute Energy
Value-based Frequency Features Category (VFFC)	Sum of Reoccurring Data Points, Sum of Reoccurring Values, Uniqueness Rate, Number of Mean Crossings
Trend-identifying Features Category (TIFC)	Aggregate Linear Trend (21), LLSR Trend (2), C3 Non-Linearity (3), Benford Correlation, AR Coefficient (2), Partial Autocorrelation, CPSD (3)
Transform-based Features Category (TFC)	FFT Coefficients (21), FFT Spectrum Features (4), CWT Coefficients (43)
Entropy-based Features Category (EFC)	Binned Entropy, Lempel-Ziv Complexity (3), Permutation Entropy (5)

Statistical Features Category (SFC): This category contains simple statistical features that can be extracted from a time series. Let x denote an ALS time series and let x_i denote the i'th sample in x, where $i \in [1, 120]$. SFC contains $min(x)$, $max(x)$, $mean(x)$, $median(x)$, $var(x)$, and $stdev(x)$, which are well-known statistical features. Here, $var(x)$ denotes the variance, and $stdev(x)$ denotes the standard deviation of x. Another well-known feature, $quantile(x, q)$, finds the value of x_i that is greater than q% of the ordered values from x. This feature was found to be statistically significant for 8 different q values, which were included in SFC. $skewness(x)$ computes the sample skewness of x calculated with the adjusted Fisher-Pearson standardized moment coefficient, mathematically defined as:

$$skewness(x) = \frac{\sqrt{n(n-1)}}{n-2} \cdot \frac{\sum_{i=1}^{n}(x_i - mean(x))/n}{stdev(x)^3}$$

where $n = |x|$.

The Variation Coefficient of x is defined as:

$$var\_coef(x) = \frac{stdev(x)}{mean(x)}$$

Another feature in this category is the Large Standard Deviation feature, which is a boolean feature that checks if $stdev(x)$ satisfies the following criteria:

$$stdev(x) > r \cdot (max(x) - min(x))$$

for a given coefficient r. The “variance larger than standard deviation" feature is also a boolean feature, which returns true if $var(x) > stdev(x)$ and false otherwise. We note that the condition $var(x) > stdev(x)$ holds if and only if $var(x) > 1$. Thus, the intuition behind this feature is to check if the variance of x (i.e., the difference between individual readings x_i versus $mean(x)$) is large or not. This feature has been used in several recent works involving time series data or signal analysis Coolen (2019); Jokar, Azzopardi & Palotti (2023); Katsidimas, Kotzakolios, Nikoletseas, Panagiotou, Timpilis & Tsakonas (Katsidimas et al.); Katsidimas, Kotzakolios, Nikoletseas, Panagiotou & Tsakonas (2022); Richard (2021).

Finally, the $symmetry(x, c)$ feature checks whether the distribution of x looks symmetric with respect to parameter c, i.e.:

$$|mean(x) - median(x)| < c \cdot (max(x) - min(x))$$

Change-based Features Category (CFC): Change-based features capture the amount of change in the ALS time series. The first feature in this category is Mean Absolute Change, mathematically defined as:

$$\frac{1}{n-1} \cdot \sum_{i=1}^{n-1} |x_{i+1} - x_i|$$

where $n = |x|$. The absolute sum of changes is equal to the above multiplied by $(n - 1)$.

The CID complexity is an estimate of time series complexity proposed in Batista, Keogh, Tataw & De Souza (2014). It is calculated as:

$$\sqrt{\sum_{i=2}^{n} (x_i - x_{i-1})^2}$$

Another feature in this category is Change Quantiles. This feature establishes fixed corridors of x_i values and then calculates the mean, median, and variance of changes within that corridor. For example, a horizontal corridor of ambient light values can be established in Figure 3.3, and then, the mean, median, and variance of changes between consecutive values $(x_{i+1} - x_i)$ within this corridor are computed.

Finally, the Sum of Values feature computes: $\sum_{i=1}^{n} x_i$ and the Absolute Energy feature computes: $\sum_{i=1}^{n} x_i^2$.

Value-based Frequency Features Category (VFFC): Features in this category capture the amount of reoccurrence (repetition) and/or uniqueness in the time series. The first feature in this category is the Sum of Reoccurring Data Points, which computes the summation of all values x_i that are observed more than once in x. In contrast, the Sum of Reoccurring Values feature computes a similar summation but counts each non-unique value once in the summation. The third feature in this category is Uniqueness Rate, defined as:

$$\frac{\text{Number of unique values in } x}{\text{Number of values in } x}$$

If all values in x are unique (i.e., there are zero reoccurring values), then Uniqueness Rate is 1.

The last feature in this category, named Number of Mean Crossings, is calculated as follows. We say that a mean crossing occurs if either: $\{x_i > mean(x)$ and $x_{i+1} < mean(x)\}$, or $\{x_i < mean(x)$ and $x_{i+1} > mean(x)\}$. Then, the Number of Mean Crossings is defined as the number of such crossings throughout the whole time series x.

Trend-Identifying Features Category (TIFC): Features in this category are used to capture linear or non-linear trends in time series. For example, Aggregate Linear Trend and LLSR Trend features calculate a linear least-squares regression model using the time series and fetch the parameters of the regression model. In contrast, C3 non-linearity was proposed in Schreiber & Schmitz (1997) as a measure of non-linearity in time series. The C3 function takes as input a lag parameter ℓ and calculates:

$$\frac{1}{n-2\ell} \sum_{i=1}^{n-2\ell} (x_{i+2\ell}) \cdot (x_{i+\ell}) \cdot (x_i)$$

Another feature in this category is the Benford Correlation, which computes the correlation between the first digit distribution of x when compared to the Newcomb-Benford's Law distribution Benford (1938); Hill (1995).

The AR Coefficient feature fits an unconditional maximum likelihood estimator of an autoregressive process to the time series. The resulting parameters (coefficients) of the autoregressive process are used as features. Relevant to the AR coefficient is the Partial Autocorrelation feature. The partial autocorrelation of a time series x equals the correlation of x_i and $x_{i-\ell}$ where ℓ is a lag parameter Box, Jenkins, Reinsel & Ljung (2015). To compute the partial autocorrelation, autoregressive models are fit to x_i and $x_{i-\ell}$, and then the correlation between them is measured.

The last feature in this category is CPSD, which estimates the cross-power spectral density of time series x Oppenheim, Schafer & Buck (1999). The power spectrum of three frequencies is used as features in our downstream models.

Transform-based Features Category (TFC): Features in this category leverage two popular transformations: Fast Fourier Transform (FFT) and Continuous Wavelet Transform (CWT). FFT is a valuable tool for detecting repeating periodic changes or seasonality within a time series. It transforms the time series from its time domain representation to the frequency domain. The Fourier coefficients of the one-dimensional discrete Fourier transform are defined as:

$$A_k = \sum_{m=0}^{n-1} a_m \cdot exp\Big(-2\pi i \frac{mk}{n} \Big)$$

for $k = 0, 1, ..., n-1$. In addition to these FFT coefficients, four FFT spectrum features are used in our attack, which correspond to the mean, variance, skew, and kurtosis of the Fourier transform spectrum.

The second transformation used in our feature set is CWT. We compute the CWT using the Ricker wavelet, mathematically defined as:

$$\frac{2}{\sqrt{3a}\pi^{\frac{1}{4}}} \cdot (1 - \frac{x^2}{a^2}) \cdot exp(-\frac{x^2}{2a^2})$$

where a is the width parameter of the wavelet function. CWT coefficients obtained after the transformation are used as part of our feature set.

Entropy-based Features Category (EFC): Our last feature category is EFC, which contains features derived from the amount of entropy in the time series x. The three features in this category are: Binned Entropy, Permutation Entropy, and Lempel-Ziv Complexity.

Binned Entropy first places the values x_i into ϕ equal-sized bins. Let p_k denote the percentage of values placed in bin k. Then, Binned Entropy is equal to:

$$- \sum_{k=0}^{min(\phi,\ len(x))} p_k \cdot \log(p_k) \cdot \mathbf{1}_{(p_k > 0)}$$

Permutation Entropy Bandt & Pompe (2002); Henry & Judge (2019) first places the values into sub-windows of length D starting every τ. Then, each sub-window of length D is replaced by its permutation such that the permutation is ordered. Finally, the frequencies of each permutation are counted, and their entropy is found:

$$-\sum_{i=1}^{D!} p_i \cdot \log_2(p_i)$$

where p_i is the frequency of permutation i.

Lempel-Ziv complexity measures the entropy of a time series by calculating its richness according to the Lempel-Ziv compression algorithm Lempel & Ziv (1976); Mitchell (2009). A time series that is more difficult to compress is said to contain higher entropy. When measuring the Lempel-Ziv complexity, the time series is first divided into a given number of bins. Then, it is converted into words with different prefixes. The number of words needed for this conversion divided by $len(x)$ yields the Lempel-Ziv complexity feature.

3.1.4 ML Model Training

LuxTrack formulates the activity inference attack as a multi-class classification problem, which is solved using supervised machine learning (ML). Let $\mathcal{D}$ denote the training dataset after feature extraction and selection. The format of $\mathcal{D}$ is: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ...\}$, such that $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,187})$ are the 187 features extracted as described in the previous section, and y_i is the class label. Since LuxTrack aims to infer activity type, y_i is the activity of the corresponding time series, i.e., one of the 8 activities listed in Section **??**. The training dataset $\mathcal{D}$ is then used to train a classifier $\mathcal{M}$, which learns to predict the activity type from the features. At test time, an unlabeled ALS time series $\mathbf{x}$ is given to LuxTrack, and $\mathcal{M}$ is able to predict the activity type of $\mathbf{x}$, i.e.: $\mathcal{M}(\mathbf{x}) \rightarrow$ activity type.

The type of classifier $\mathcal{M}$ plays an important role in determining the accuracy of the resulting attack model. When implementing LuxTrack, we experimented with diverse types of linear and non-linear classifiers. In the paper, we report results with six classifier types: Naive Bayes (NB), k-nearest neighbor (K-NN), decision trees (DT), Support Vector Machine (SVM), neural networks (NN), and gradient-boosted trees (GBT). Each classifier has different hyperparameters, e.g., number of neighbors in K-NN, layer structure in NN, number of trees and max depth in GBT. Each classifier is tuned individually to achieve high accuracy by selecting its hyperparameters using grid search and 10-fold cross validation.

3.2 Subject Inference Attack

3.2.1 Dataset

In our exploration of datasets dedicated to fall detection, we identified more than ten distinct collections obtained from individuals. Each of these datasets serves the primary objective of facilitating fall detection algorithms. After careful consideration, we opt for *SisFall* dataset due to its comprehensive recording of activities and subjects, facilitated through the use of a wearable device positioned around the waist.

SisFall dataset comprises recordings from 38 subjects, encompassing 23 young individuals and 15 elderly participants. Each subject undertakes 5 trials, engaging in 19 different types of safe activities referred to as Activities of Daily Living (ADL), as well as 15 types of activities culminating in falls (Falls). Notably, elderly subjects exclusively partake in ADL activities, with one exception—a judo expert elderly man who performs both ADL and Falls activities, akin to the young subjects. Falls activity trials are standardized at a duration of 15 seconds each, while ADL trials vary, lasting 12 seconds (for 12 activities), 20 seconds (for 4 activities), or 25 seconds (for 3 activities).

For the purpose of our studies, we selectively focus on recordings from the 23 young participants and the singular elderly participant engaged in ADLs. Consequently, we exclude recordings from other elderly participants and all recordings associated with Falls activities, regardless of the subject's age. We focus exclusively on ADL recordings to study subject inference in routine, non-emergency scenarios. Fall activities, being brief and structured, are less suitable for learning distinctive motion patterns across individuals. Additionally, we exclude ADL recordings from most elderly participants to avoid age-related variation and class imbalance, which could affect model consistency. The only exception is one elderly subject who performed all ADLs, making their data compatible with the rest of the cohort.

SisFall dataset was recorded utilizing a self-developed embedded device affixed to the participants' waist. This device incorporates a Kinets MKL25Z128VLK4 microcontroller, an Analog Devices ADXL345 accelerometer, a Freescale MMA8451Q accelerometer, an ITG3200 gyroscope, an SD card for recording, and a 1000 mA/h generic battery. All over this article, we call ADXL345 sensor as Acc1, MMA8451Q as Acc2 and ITG3200 as Gyro for simplicity. All tests were performed with the original frequency sample of 200 Hz, and three values from axis x, y and z are recorded at each time step. So, *SisFall* has 9 attributes in its recordings which are treated as columns of the dataset. This instrumentation ensures a comprehensive capture of motor signals, forming the basis for our subsequent analyses in addressing privacy concerns associated with the dataset.

3.2.2 Feature Extraction and Selection

In our research, the tsfresh library (Christ, Braun, Neuffer & Kempa-Liehr, 2018) serves as a valuable tool for feature extraction from the dataset, generating a substantial 618 non-null values features for each attribute in the input dataset. Each feature is calculated through a feature calculator given specific input values. While some feature calculators produce only one feature (like mean and standard deviation), there are some others which calculate more than one features (like fft coefficient). With three sensors, each recording x, y, and z values at time steps from the user's waist, a total of 5562 features are extracted for the 9 attributes present in *SisFall* dataset. This detailed feature set becomes integral to our subsequent machine learning model training endeavors. Our approach involves training machine learning models on various portions of the dataset, allowing for detailed exploration. From the 5562 total features, we strategically select 156 most significant (with pValue <= 0.05), choosing those with the lowest pValue among their counterparts. Feature selection is carried out on various subsets of data chosen for training ML models, thus 156 most significant features from different portion of data might have some common and some different features from each others. Details about the selected features are provided in their related sub sections.

A total of 50 feature calculators have been used during our experiments in this research, and here we will briefly mention the definition and performance of each one.

Statistical Features: Some of selected features are just simple statistical features that can be calculated from a trial record. Let's say x is a time series of a trial recorded from a user doing a specific activity. The time_reversal_asymmetry_statistic(x, lag) and symmetry_looking(x, c) features check whether the distribution of x looks symmetric. The kurtosis(x) and skewness(x) uses adjusted Fisher-Pearson standardized moment coefficient G2 to compute the sample kurtosis and skewness of time series x. Large Standard Deviation feature, which is a boolean feature and for a given coefficient (r) checks if standard deviation of time series is greater than r times the its mean. The Variation Coefficient which is defined as standard deviation of time series over its mean value. The Variance, standard_deviation, mean, median, quantile and maximum are another well-known features that quantile(x, q), finds a value in time series x that is greater than q% of the ordered values.

Change-based Features: This set of features are from those which capture the amount of change in a trial. The first feature in this category is Change Quantiles. This feature establishes fixed corridors of time series values, and then calculates the mean, median, and variance of changes within that corridor. sum_values(x) calculates the sum over values of a time series and absolute_sum_of_changes(x) computes the sum over the absolute value of consecutive changes in the series x. The mean_abs_change(x) returns the mean over the absolute differences between subsequent time series values and abs_energy(x) feature computes sum of squares of a time series values. The energy_ratio_by_chunks(x, param) calculates sum of squares of chunk i out of N chunks expressed as a ratio with the sum of squares over the whole series. The cid_ce(x, normalize) is another feature in this set which is an estimate of time series complexity.

Value-based Frequency Features: Features grouped in this set capture the amount of reoccurrence (repetition) and/or uniqueness in the time series of trials. The first feature in this set is ratio_beyond_r_sigma(r, x) which computed as one of the most significant features from most of sub-data. This feature computes the ratio of values that are more than r * standard deviation of a time series(x), so r times sigma, away from the mean of x. Sum of Reoccurring Data Points, which computes the summation of all values of a time series that are observed more than once. In contrast, the Sum of Reoccurring Values feature computes a similar summation, but counts each non-unique value once in the summation. The percentage of reoccurring values to all values returns the percentage of values that are present in the time series more than once. This means the percentage is normalized to the number of unique values. The range_count(x, min, max) is the next feature in this group and counts observed values within the interval [min, max]. While count_below(x, t) feature calculates the percentage of values in x that are lower than t, the count_above(x, t) returns the percentage of values in x that are higher than t. The value_count(x, value) counts occurrences of value in time series x and number_crossing_m(x, m) calculates the number of crossings of x on m. A crossing is defined as two sequential values where the first value is lower than m and the next is greater, or vice-versa. The last feature in this category is number_peaks(x, n) which computes the number of peaks of at least support n in the time series x. A peak of support n is defined as a subsequence of x where a value occurs, which is bigger than its n neighbours to the left and to the right.

Trend-Identifying Features: Features in this set are used to capture linear or non-linear trends in trials. The linear_trend(x, params), for example, calculates a linear least-squares regression for values of the time series of a trial that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one. The Aggregate Linear Trend and LLSR Trend features calculate a linear least-squares regression model using the time series and fetch the parameters of the regression model. In contrast, C3 non-linearity was proposed as a measure of non-linearity in time series. AR Coefficient feature fits an unconditional maximum likelihood estimator of an autoregressive process to the time series. Resulting parameters (coefficients) of the autoregressive process are used as features. Relevant to the AR coefficient are the well-known Autocorrelation and Partial Autocorrelation features. The agg_autocorrelation(x, param) is another similar feature and calculates the value of an aggregation function f_agg (e.g. variance or mean) over the autocorrelation for different lags. Another feature in this set is SPKT Welch Density (CPSD), which estimates the cross-power spectral density of time series x. The power spectrum of three frequencies is used as features in our downstream models. The friedrich_coefficients(x, param) feature computes coefficients of polynomial h(x), which has been fitted to the deterministic dynamics of langevin model and max_langevin_fixed_point(x, r, m) returns the largest fixed point of dynamics estimated from polynomial h(x), which has been fitted to the deterministic dynamics of Langevin model. Benford Correlation, which computes the correlation between the first digit distribution of x when compared to the Newcomb-Benford's Law distribution is the next feature in this group. The Augmented Dickey Fuller test is a hypothesis test which checks whether a unit root is present in a time series sample. The last feature of the group calculates the value of the respective test statistic.

Transform-based Features: This set has 3 featues with similar functionality on transformation. Fast Fourier Transform (FFT), its aggregated form and Continuous Wavelet Transform (CWT) are popular transformations. FFT is a valuable tool for detecting repeating periodic changes or seasonality within a time series. It transforms the time series from its time domain representation to the frequency domain. The aggregated fft computes the spectral centroid (mean), variance, skew, and kurtosis of the absolute fourier transform spectrum. CWT coefficients calculates a continuous wavelet transform for the Ricker wavelet, also known as the Mexican hat wavelet taking "widths", "coeff" and "w". The feature calculator takes all the different widths arrays and then calculates the cwt one time for each different width array. Then the values for the different coefficient for "coeff" and width "w" are returned.

Entropy-based Features: Our last set of features contains those derived from the amount of entropy in the time series of a given trial x. The feature calculators in this set are: *permutation_entropy*, *approximate_entropy*, *sample_entropy*, *binned_entropy*, and *fourier_entropy.*

Permutation Entropy (Bandt & Pompe, 2002; Watt & Politi, 2019) first places the values into sub-windows of length D starting every θ. Then, each sub-window of length D is replaced by its permutation such that the permutation is ordered. Finally, the frequencies of each permutation are counted, and their entropy is computed:

$$PE_D = -\sum_{i=1}^{D!} p_i \log_2(p_i)$$

where p_i is the frequency of permutation i.

Approximate Entropy (Richman & Moorman, 2000; Yentes, Hunt, Schmid, Kaipust, McGrath & Stergiou, 2013) measures the regularity and unpredictability of fluctuations in a time series. It is computed by comparing sequences of length m within the time series and evaluating their similarity within a tolerance r. For longer time series (e.g., $N > 2000$), this method yields stable results. It provides a scalar value that reflects the system's complexity — lower values correspond to more predictable patterns, while higher values indicate greater irregularity.

Sample Entropy (Richman & Moorman, 2000) is a refinement of Approximate Entropy that improves consistency by avoiding self-matching in the analysis. It estimates the negative natural logarithm of the conditional probability that sequences of length m that match within tolerance r will remain similar at the next point. Compared to Approximate Entropy, Sample Entropy is less biased and more reliable, especially for shorter time series.

Binned Entropy first places the values x_i into ϕ equal-sized bins. Let p_k denote the percentage of values placed in bin k. Then, Binned Entropy is equal to:

$$-\sum_{k=0}^{\min(\phi,\ len(x))} p_k \cdot \log(p_k) \cdot \mathbf{1}_{(p_k>0)}$$

Fourier Entropy calculates the entropy of the power spectral density (PSD) of the signal using Welch's method (Bruce Land, Bruce Land; The Scipy community, The Scipy community). It first estimates the PSD of the time series, discretizes the resulting spectral values into bins, and then computes Binned Entropy over the normalized power distribution. This feature captures the complexity of the signal in the frequency domain, offering complementary information to the time-domain entropy measures.

3.2.3 Subject Inference Attack Approaches

In this section, we explore approaches to infer the subject performing a trial based on sensor data. The first method directly predicts the subject from the entire dataset, using machine learning models trained on features extracted from sensor data across 19 activities. After feature extraction, we select significant features and train classifiers, such as decision trees, XGBoost, and LightGBM, to identify the subject. The second approach, more closely resembling real-world scenarios, first predicts the activity type and then narrows the focus to the subset of data corresponding to that activity to infer the subject. This method assumes that, in practice, activity information would be available, but subject identity would be hidden. Both approaches rely on careful feature selection and model tuning to achieve optimal prediction accuracy.

3.2.3.1 First cut solution

In the first solution, we go directly to subject prediction from the whole dataset. This approach is the simplest solution comes to the mind and can be interpreted as if the attacking algorithm tries to predict the performer of a trial regardless of knowing the activity this trial comes from. In this approach, we give 4 trials out of 5 recorded from each subject from 19 activities to our ML models as training data to predict the subject of trial from the 156 features extracted of each trial. The schematic of this approach shows the process in detail (see figure 3.4). Let's express our dataset as $D\,(\text{Subject, Activity, Trial, Sensor, Axis})$ where D gives us values of an attribute coming from one of three sensors in one of axes x, y or z. The *Sensor* and *Axis* shows these input well. For our dataset, *Subject* input specifies one of 24 subjects' Subject_Id who performed the given *Trial* from 4 trials any subject performed for each of 19 possible activities. For our sensors with 200 Hz sampling frequency, the trial Trl_k from a 20-second lasted activity like Act_j who performed by Sub_i contains 4000 samples of 9 sensor values (3 axes of 3 sensors). Therefore, an attribute of this trial which is accessed by $D\,(\text{Sub}_i,\ \text{Act}_j, \text{Trl}_k,\ \text{Sns}_m, \text{Axs}_n)$ will return a list of 4000 values.

In the first step, the meaningful features are calculated from attributes of all trials. There are fixed 618 features to be extracted from each 9 attributes of a trial. They can be appended to a single list to form 9×618 long array of total features extracted from that trial. When the process continues for all available trials in our training dataset, the row-by-row union of features arrays generated from trails makes a matrix of features which we is returned at the end. The algorithmic explanation (see pseudocode no) below shows the process in more clear way. FE is the function that calculates the fixed number of features for the specified attribute of a trial.

```
                                                        ▷ FE: Feature Extraction Function
function FE($D\,(\text{Subject, Activity, Trial, Sensor, Axis})$)
    Calculate 618 fixed feature values for each sensor axis of a trial
    Return them as a list of values
```

```
end function

$feature\_matrix \leftarrow []$                                        ▷ Clear lists
$feature\_values \leftarrow []$
$features \leftarrow []$
for each Sub in Subject list do
    for each Act in Activity list do
        for each Trl in Trial list do
            $feature\_values \leftarrow []$
            for each Sns in Sensor list do
                for each Axs in Axis list do
                    $features \leftarrow FE(D(\text{Sub, Act,Trl, Sns, Axs}))$
                    Append features list to feature_values list
                end for
                Append feature_values list to feature_matrix list
            end for
        end for
    end for
    Return feature_matrix
end for
```

Of course many of those features extracted for trials won't have noticeable effect on trials classification. Thus, the next step of process is selecting some of the most significant features from those many features extracted for trials which have the highest impact on trials classification based on *subject_id*. pValue is the metric we use to measure the significance of features in classification, so that the features dataset ($features\_matrix$) extracted from the training trials dataset is given to Feature Selection function (*FS*) to have the required correlation between each feature and the label column (here *subject_id*) calculated as pValue of each feature. The features are then sorted from the most significant ones to the less by descending order of pValues. Since the smaller pValue is, the highest impact the feature has in classification. We then select only top n=156 features sorted (see pseudocode no).

```
                                              ▷ FS: Feature Selection Function
function FS(Data, Label)
    Calculate the pValue of each feature column in Data relative to the Label column
    Keep only the features column with pValue <= 0.05
    Select and return top 156 features columns with lowest pValues
```

```
end function

$selected\_feature\_matrix \leftarrow []$                ▷ Clear list
$selected\_feature\_matrix \leftarrow FS(\text{feature\_matrix, Subject\_Id})$
```

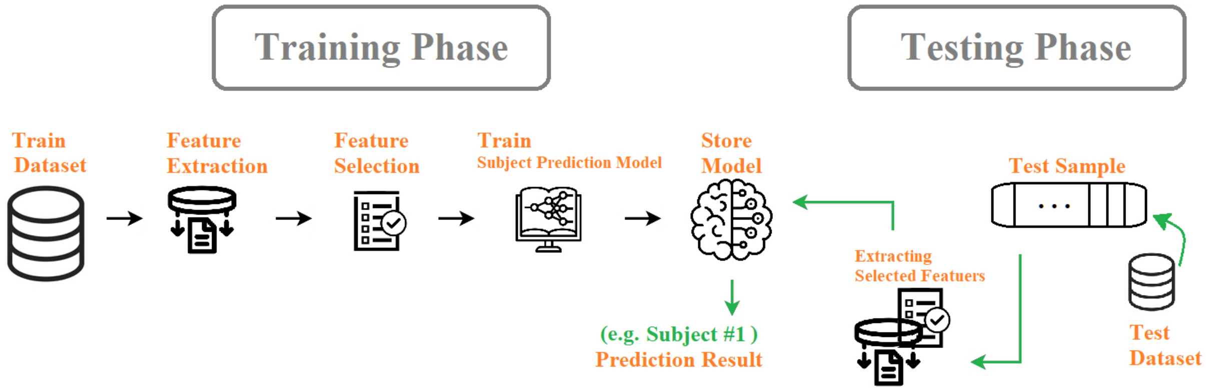


Figure 3.4 Subject Prediction Schematic in our First cut solution

In the subsequent step, we define the subject prediction attack as a multiclass classification problem, addressed through supervised machine learning (ML) techniques. Let $\mathcal{D}$ represent the training dataset after the feature extraction and selection phase (previously referred to as selected_feature_matrix). The structure of $\mathcal{D}$ is given by: $\mathcal{D} = \{(\mathbf{x}_1, y), (\mathbf{x}_2, y), ...\}$, such that $\mathbf{x}_i = (x_1, x_2, ..., x_{156})$ are feature vectors each consists of 156 extracted and selected features, and y denotes the class label. In this context, since the goal is to infer the subject identity, y corresponds to the *subject_id* indicating the individual who performed each trial. This training dataset $\mathcal{D}$ is then used to train a classifier $\mathcal{M}$, which learns to map feature vectors to their corresponding subjects. During inference, when an unlabeled trial instance $\mathbf{x}$ is presented, the classifier $\mathcal{M}$ outputs its predicted subject identity, i.e., $\mathcal{M}(\mathbf{x}) \rightarrow \text{subject}$.

The choice of classifier $\mathcal{M}$ significantly influences the effectiveness of the attack model. In our implementation, we explored various classifier architectures and report our results using four tree-based models: decision trees (DT), gradient-boosted trees (GBT), extreme gradient boosting (XGBoost), and light gradient boosting (LightGBM). Each classifier was individually tuned through grid search combined with 4-fold cross-validation, allowing us to optimize their hyperparameters and maximize prediction accuracy.

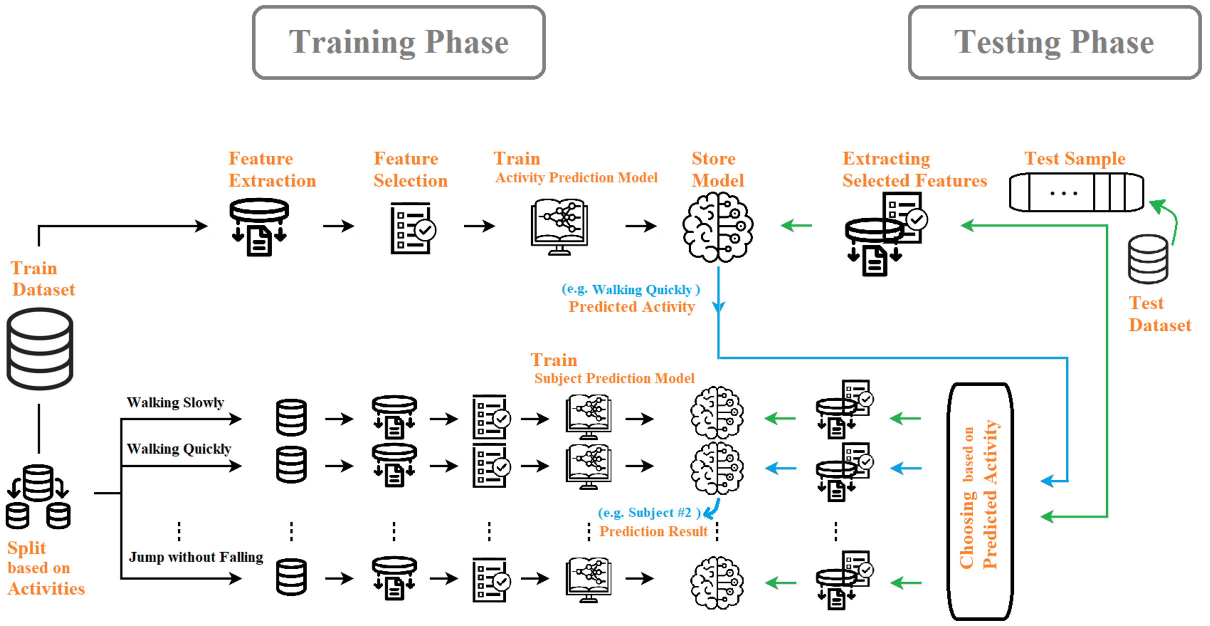


Figure 3.5 Subject Prediction Schematic in our Proposed solution

3.2.3.2 Proposed solution

In this approach we do not predict the subject directly, but to predict the activity of trials first. In this solution, the attacking model tries to predict the activity from which the trial was recorded without needing to know the subject id of trial. Thus this solution is closer to the reality. Because this dataset and the other similar ones are recorded mainly for fall detection purposes, thus in real scenarios, in order to keep the privacy, the information about subjects who performed the trials will be removed before releasing the dataset, and data will be sent to the clients with the activity code. This means that the attacking system will be aware of the activity the trials come while accessing the data, but it does not have access to the subjects who made the trials. Therefore, it will probabily train a model to predict the activity on the entire data set. After predicting the attempted activity, train a model to predict the subject of activity only on the subset of data related to the pre-predicted activity. See the schematic of this approach in the figure 3.5.

We formulate both the activity and subject inference tasks as multiclass classification problems, solved through supervised machine learning (ML) methods. Let $\mathcal{D}$ denote the dataset constructed after applying feature extraction and selection. The dataset is structured as: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$ where each instance $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,156})$ represents a 156-dimensional feature vector, and y_i indicates the associated class label. These features are extracted as described in the preceding section. When the objective is to predict activities, y_i corresponds to the activity code associated with the respective trial—specifically, one of four labeled trials out of five recorded for each of the 19 distinct activities performed by 24 subjects. The classifier $\mathcal{M}$ is trained on the dataset $\mathcal{D}$ to learn the mapping between feature vectors and activity types. During inference, when an unlabeled trial vector $\mathbf{x}$ is given to the system, $\mathcal{M}$ predicts the activity type as $\mathcal{M}(\mathbf{x}) \rightarrow$ activity type.

A similar process is followed when the goal is to predict subjects; however, in this case, the subject is not inferred directly from the entire dataset. Instead, the system first determines the activity type of the trial using $\mathcal{M}$ as described above. Then, subject inference is performed using only the subset of the dataset corresponding to the predicted activity. For instance, if the predicted activity type for an unlabeled trial is **D12** ("sitting a moment, lying slowly, wait a moment, and sit again"), which is one of the 19 defined activities (**D01** to **D19**), the system performs subject classification using only the 1/19 portion of the dataset that includes trials labeled as **D12**. At this stage, a separate supervised classifier $\mathcal{N}$ is trained to infer the subject identity from the restricted feature set, i.e., $\mathcal{N}(\mathbf{x}) \rightarrow$ subject id.

Choosing suitable models for $\mathcal{M}$ and $\mathcal{N}$ is essential to ensure high performance in activity and subject prediction. To this end, we explored both linear and non-linear classification techniques. The experimental results reported in this work are based on four widely-used tree-based classifiers: decision trees (DT), gradient-boosted trees (GBT), extreme gradient boosting (XGBoost), and light gradient boosting (LightGBM). Each model are fine-tuned independently using grid search along with 4-fold cross-validation to identify optimal hyperparameters and enhance predictive accuracy.

4. ATTACKS RESULTS AND DISCUSSIONS

This section presents the experimental evaluation and analysis of the proposed attacks. We organize our results and discussions into two main parts. The first part focuses on the ALS-based Activity Inference Attack, where we analyze the performance of LuxTrack under various conditions, compare it with existing baselines, evaluate robustness, and assess generalization through multiple experimental settings. The second part addresses the Subject Inference Attack, examining both a preliminary baseline solution and our proposed refined approach. Through this structure, we aim to highlight the effectiveness of our attacks as well as the insights gained from their detailed performance analyses.

4.1 ALS-based Activity Inference Attack Results and Discussions

We evaluate the effectiveness of LuxTrack using the dataset we collected. All results are reported with 10-fold cross validation. We use well-known metrics to evaluate the success rate of LuxTrack: accuracy, precision, recall, and F1 score.

4.1.1 Comparison with LightSpy and Raw ALS Data

First, in order to demonstrate the improvement achieved by LuxTrack, we perform a comparison between LuxTrack, the state-of-the-art LightSpy attack Chakraborty et al. (2017), and the use of raw ALS time series recordings to train ML models (instead of feature extraction and selection). We note that the methodology used in LuxTrack differs from LightSpy in terms of the features extracted and the ML models used. Furthermore, the comparison between LuxTrack and raw ALS recordings aims to show the benefit of LuxTrack's feature extraction step rather than feeding ALS recordings to ML models directly (without feature extraction).

The results of this comparison are shown in Table 4.1. The accuracy and F1 scores of each model are reported after the models' hyperparameters are optimized in each setting. Since there are 8 total activities, the accuracy of a random guess is 12.5%. In general, models achieve substantially higher accuracy compared to a random guess, which

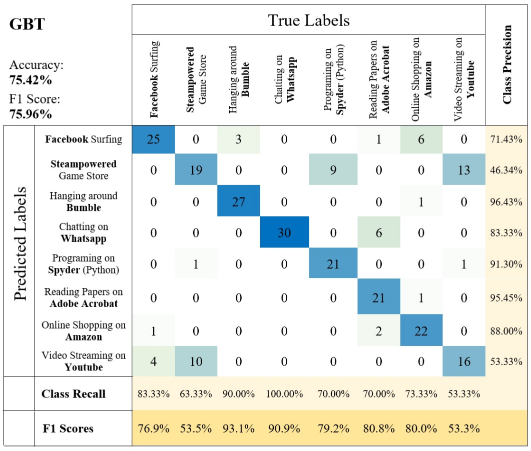
Table 4.1 Accuracy and F1 scores of our LuxTrack attack versus LightSpy versus using raw ALS time series recordings in classification (without feature extraction). Results show that LuxTrack outperforms its most relevant competitor, LightSpy, by roughly 20% in terms of accuracy and F1 score.

	Raw ALS		**LightSpy**		**LuxTrack (Our Attack)**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	20.00%	16.26%	51.67%	47.05%	46.25%	40.00%
K-NN	52.08%	49.66%	60.00%	58.01%	70.00%	68.99%
DT	35.83%	33.30%	57.92%	58.59%	62.50%	64.25%
SVM	33.33%	27.65%	45.42%	39.96%	61.25%	57.68%
NN	52.08%	50.80%	-	-	79.58%	79.56%
GBT	54.17%	53.75%	-	-	75.42%	75.96%
Best	54.17%	53.75%	60.00%	58.59%	79.58%	79.56%

indicates the plausibility of inferring user activity from ALS readings. We observe that NB and SVM models usually yield lower accuracy and F1 scores, whereas NN and GBT models' accuracy and F1 scores are generally higher.

The best-performing model for raw ALS recordings is GBT, which achieves 54.17% accuracy and 53.75% F1 score. In contrast, the best-performing models are K-NN and DT in terms of accuracy and F1 score, respectively. The accuracy and F1 score of the best-performing models for LightSpy are higher than the accuracy and F1 score for raw ALS. Finally, the highest accuracy and F1 scores are achieved using LuxTrack. The best-performing model for LuxTrack is NN with 79.58% accuracy and 79.56% F1 score, which are roughly 20% higher than the accuracy and F1 score achieved by LightSpy. This demonstrates that LuxTrack clearly outperforms its closest competitor.

When we compare LightSpy against LuxTrack with respect to each model individually, we observe that LightSpy performs better than LuxTrack only in terms of NB. For the remaining models (K-NN, DT, SVM), LuxTrack consistently yields higher accuracy and F1 score. Thus, we believe that the superiority of LuxTrack compared to LightSpy is not because of the model type, but rather because of the set of features. Only 8 features were used in LightSpy, which are: range, standard deviation, mean absolute derivative, mean crossing rate, skewness, entropy, high-frequency energy ratio, and spectral mean Chakraborty et al. (2017). LuxTrack also uses some of these features, but in addition, we include many other informative and statistically significant features which enable LuxTrack to achieve significantly higher attack effectiveness than LightSpy.



GBT

Accuracy: **75.42%**

F1 Score: **75.96%**

		True Labels								
		Facebook Surfing	**Steampowered** Game Store	Hanging around **Bumble**	Chatting on **Whatsapp**	Programing on **Spyder** (Python)	Reading Papers on **Adobe Acrobat**	Online Shopping on **Amazon**	Video Streaming on **Youtube**	**Class Precision**
Predicted Labels	**Facebook** Surfing	25	0	3	0	0	1	6	0	71.43%
	Steampowered Game Store	0	19	0	0	9	0	0	13	46.34%
	Hanging around **Bumble**	0	0	27	0	0	0	1	0	96.43%
	Chatting on **Whatsapp**	0	0	0	30	0	6	0	0	83.33%
	Programing on **Spyder** (Python)	0	1	0	0	21	0	0	1	91.30%
	Reading Papers on **Adobe Acrobat**	0	0	0	0	0	21	1	0	95.45%
	Online Shopping on **Amazon**	1	0	0	0	0	2	22	0	88.00%
	Video Streaming on **Youtube**	4	10	0	0	0	0	0	16	53.33%
	Class Recall	83.33%	63.33%	90.00%	100.00%	70.00%	70.00%	73.33%	53.33%	
	F1 Scores	76.9%	53.5%	93.1%	90.9%	79.2%	80.8%	80.0%	53.3%	

(a) GBT

NN

Accuracy: **79.58%**

F1-Score: **79.56%**

		True Labels								
		Facebook Surfing	**Steampowered** Game Store	Hanging around **Bumble**	Chatting on **Whatsapp**	Programing on **Spyder** (Python)	Reading Papers on **Adobe Acrobat**	Online Shopping on **Amazon**	Video Streaming on **Youtube**	**Class Precision**
Predicted Labels	**Facebook** Surfing	18	0	2	0	0	0	4	0	75.00%
	Steampowered Game Store	1	26	0	0	2	0	0	14	60.47%
	Hanging around **Bumble**	0	0	28	0	0	1	2	0	90.32%
	Chatting on **Whatsapp**	0	0	0	28	0	3	0	0	90.32%
	Programing on **Spyder** (Python)	1	1	0	0	28	0	0	1	90.32%
	Reading Papers on **Adobe Acrobat**	0	0	0	1	0	24	0	0	96.00%
	Online Shopping on **Amazon**	0	0	0	1	0	2	24	0	88.89%
	Video Streaming on **Youtube**	10	3	0	0	0	0	0	15	53.57%
	Class Recall	60.00%	86.67%	93.33%	93.33%	93.33%	80.00%	80.00%	50.00%	
	F1 Scores	66.7%	71.2%	91.8%	91.8%	91.8%	87.3%	84.2%	51.7%	

(b) NN

Figure 4.1 Confusion matrices of GBT and NN models obtained after executing LuxTrack.

4.1.2 Analysis of Confusion Matrices

In Figure 4.1, we illustrate the confusion matrices of the GBT and NN models after executing LuxTrack. We choose to analyze the confusion matrices of GBT and NN models rather than other models, since GBT and NN yield the highest accuracy and F1 score.

We observe that the confusion matrices of GBT and NN agree in several aspects. For example, according to both models, the activity types (classes) with the lowest precision are Steampowered game store and video streaming on Youtube. In particular, both models are likely to confuse video streaming on Youtube with Steampowered game store activity. Programming on Spyder is also likely to be confused with Steampowered game store. The likely reason behind these confusions is because Steampowered, Spyder, and Youtube videos typically have dark background colors, resulting in a low amount of light being emitted from the laptop screen. In contrast, the recall and F1 scores of Bumble and Whatsapp activities are high for both NN and GBT. This is because applications like Whatsapp emit consistent amounts of light throughout the activity without much fluctuations, which causes them to be more easily inferred.

In general, other than three relatively more challenging activities (which are Facebook surfing, Steampowered game store, and video streaming on Youtube), the F1 scores of each class are higher than 80-90% for both NN and GBT. This shows that many laptop activities have unique light emission characteristics, which can allow them to be inferred by adversaries using an ALS eavesdropping attack like ours.

Table 4.2 Analysis of feature importance in terms of feature categories. For each rank range (e.g., top 10, 11-20, 21-30), the table shows what percentage of features within that range fall under each feature category. For example, 50% of features in top 10 fall under SFC category, 10% of features in top 10 fall under CFC category, 60% of features in 11-20 fall under TIFC category, etc.

Feature Importance Rank	Percentage of Features Within Rank Under Each Category					
	SFC	CFC	VFFC	TIFC	TFC	EFC
Top 10	50%	10%	10%	10%	20%	0%
11 - 20	20%	10%	10%	60%	0%	0%
21 - 30	10%	0%	0%	50%	30%	10%
31 - 40	10%	60%	0%	10%	20%	0%
41 - 50	10%	0%	0%	0%	90%	0%
51 - 75	8%	8%	0%	4%	72%	8%
76 - 100	16%	40%	4%	0%	32%	8%
101 - 125	8%	64%	0%	0%	16%	12%
126 - 150	12%	32%	0%	28%	28%	0%
151 - 187	13.5%	8.1%	2.7%	32.5%	40.5%	2.7%

4.1.3 Feature Importance Analysis

Next, we analyze the feature importance of each feature used in LuxTrack's models. We used p-values of features to perform this analysis. The lower the p-value of a feature, the more important role it plays in prediction. We computed the p-value of each feature and then ranked features (from 1 to 187) according to their p-values, with the feature in 1st ranking being the most important. Considering the high number of features, instead of presenting the full feature importance ranking, we divided the ranking into ranges such as top 10, rank 11-20, rank 21-30, .., rank 151-187. For each range, we found what percentage of features within that range fall under each feature category (according to the categorization in Table 3.1). The results are given in Table 4.2.

Among all categories, SFC has the highest number of features in the top 10 ranks, with 5 out of 10 features in the top 10 belonging to the SFC category. In addition, 2 SFC features are ranked between 11-20. Quantile is the most important feature, followed by the Minimum, Mean, and Median features. These results indicate that simple aggregate statistics such as quantile, minimum, mean, and median can indeed serve as good predictors of user activity from ALS.

Following SFC, the TIFC category also plays a significant role in our classification models. It contains one feature among the top 10, 6 features in the range 11-20, and 5 features in the range 21-30. The most important features in the TIFC category include Aggregate Linear Trend, C3 Non-Linearity, LLSR Trend, and Benford Correlation (in this order). There also exist other significant features in this category, but their importance rankings are comparatively lower.

After SFC and TIFC, we have TFC, CFC, and VFFC categories. In the TFC category, FFT coefficients and FFT spectrum features constitute features that are ranked in the top 40. In contrast, CWT coefficients rank lower, typically within the 41-50 range. In the CFC category, Sum of Values and Absolute Energy features are the highest-ranking features, which are ranked 10th and 12th, respectively. They are followed by the Change Quantiles feature, Mean Absolute Change feature, Absolute Sum of Changes feature, and CID complexity feature, respectively. In the VFFC category, the Sum of Reoccurring Data Points feature is ranked 5th, and the Sum of Reoccurring Values feature is ranked within the top 20. In contrast, the Uniqueness Rate and Number of Mean Crossings features are ranked lower.

Finally, features in the EFC category rank lowest among other categories. The Binned Entropy feature ranks 22nd, but the remaining features in the EFC category are ranked between 51-178. Considering these results, we can conclude that entropy-related features contribute less than other features to our LuxTrack attack.

4.1.4 Attack Robustness Evaluation

In this section, we perform a broader attack robustness evaluation to validate the effectiveness of LuxTrack in varying conditions, such as varying display technologies, display settings, ambient light conditions, smartphone models, and spatial relationships between the ALS and the display. For consistency, the same participants, activities, and durations were used as in Section 3.1.2. The original ML models were *not* retrained, i.e., the original models were tested with different display technologies, display settings, etc. This enables us to study the generality and transferability of LuxTrack's models to various conditions.

Impact of different display technologies. The laptop model used in our original data collection and the experiments reported thus far was a Lenovo E480 equipped with a TN display. Here, we measure the impact of using a different laptop (Asus UX430U) equipped with an IPS display. The results are reported in Table 4.3.

We observe from Table 4.3 that LuxTrack is effective on an IPS display, reaching close

Table 4.3 Accuracy and F1 scores of LuxTrack on an Asus laptop with IPS display.

	LuxTrack (IPS Display)	
Model	**Accuracy**	**F1 Score**
NB	52.08%	47.36%
K-NN	71.25%	70.56%
DT	61.67%	61.58%
SVM	52.08%	49.01%
NN	74.17%	73.41%
GBT	71.25%	70.96%
Best	74.17%	73.41%

to 74% accuracy and F1 score using an NN model. Similar to Table 4.1 (in which an HD display was used), NN and GBT models provide higher accuracy and F1 scores compared to other models on an IPS display as well. In Table 4.3, NN and GBT are followed closely by K-NN, and the remaining models are inferior. Also, comparing Tables 4.1 and 4.3, we observe that NB and K-NN perform slightly better in Table 4.3 as opposed to Table 4.1, whereas the remaining models perform slightly better in Table 4.1. Considering all results, we conclude that the change from a TN display to an IPS display yields small drops in accuracy and F1 score, but LuxTrack is still highly effective for an IPS display, achieving 74.17% accuracy.

Table 4.4 Accuracy and F1 scores of LuxTrack under varying brightness levels.

	100% Brightness		**80% Brightness**		**60% Brightness**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	46.25%	40.00%	42.08%	35.61%	15.00%	9.10%
K-NN	70.00%	68.99%	74.58%	74.13%	25.42%	18.34%
DT	62.50%	64.25%	66.25%	63.86%	27.50%	19.65%
SVM	61.25%	57.68%	53.33%	51.94%	22.08%	13.76%
NN	79.58%	79.56%	75.83%	76.40%	30.83%	18.81%
GBT	75.42%	75.96%	71.25%	70.34%	29.17%	19.84%
Best	79.58%	79.56%	75.83%	76.40%	30.83%	19.84%

Impact of different display settings. Next, we keep the display technology constant by using the Lenovo E480 but vary two of the display settings: brightness and gamma value. Brightness can be controlled using percentages (e.g., 100%, 90%, 80%). For the gamma value, we test three different settings: maximum gamma, medium gamma, and minimum gamma. In the previously reported experiments, the default settings of 100%

Table 4.5 Accuracy and F1 scores of LuxTrack under varying gamma settings.

	Max Gamma		**Medium Gamma**		**Min Gamma**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	63.75%	59.16%	46.25%	40.00%	35.42%	32.13%
K-NN	80.42%	79.61%	70.00%	68.99%	72.50%	71.14%
DT	69.17%	69.36%	62.50%	64.25%	54.17%	49.31%
SVM	69.17%	65.04%	61.25%	57.68%	46.67%	43.71%
NN	83.33%	83.29%	79.58%	79.56%	77.50%	77.89%
GBT	82.08%	82.09%	75.42%	75.96%	71.25%	70.94%
Best	83.33%	83.29%	79.58%	79.56%	77.50%	77.89%

brightness and medium gamma were used. Note that these are default settings and they are expected to be in line with an everyday laptop user. In the next sets of experiments, we vary the brightness percentage and gamma setting individually and report their impacts on attack effectiveness.

In Table 4.4, we keep the default value of gamma (medium) and vary brightness between 100% and 60%. When brightness is reduced from 100% to 80%, the accuracy of LuxTrack decreases by only 4%. However, when brightness is further reduced to 60%, the accuracy of LuxTrack decreases to 30.83%. These results indicate that attack effectiveness is indeed sensitive to brightness, which is an intuitive result due to the fact that when brightness is 100%, there is more light emitted from the display, which causes the recorded lux values of different activities to be high and diverse (as exemplified in Figure 3.3). With 80% brightness, the lux values we recorded were closer to one another (e.g., between 9-42 lux), but differences between activities were still substantial. However, with 60% brightness, the lux values we recorded were all within a small range (e.g., between 8-20 lux for all activities), and therefore, it becomes difficult to distinguish the light emission patterns of an activity from the others. Therefore, we can conclude that higher brightness yields higher attack effectiveness.

In Table 4.5, we keep the default value of brightness (100%) and vary the gamma. The display's gamma controls the pixels' luminance, affecting the appearance of darker areas on the screen. Lower gamma makes shadows look brighter, whereas higher gamma makes it harder to see details within shadows. LuxTrack achieves the highest accuracy and F1 score (around 83%) when gamma is maximized. When gamma is at its medium setting, the accuracy and F1 score of LuxTrack decrease to around 80%. Finally, when gamma is at its minimum setting, the accuracy and F1 score of LuxTrack decrease to around 77-78%. While changing the gamma value does impact attack effectiveness, it does not

Table 4.6 Accuracy and F1 scores of LuxTrack under two alternative ambient light conditions: reduced light (the room is illuminated by a distant bedside lamp) and daylight (the room is illuminated by sunlight during day hours).

	Reduced Light		**Daylight**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	38.33%	34.41%	22.08%	15.34%
K-NN	51.67%	46.85%	35.00%	35.26%
DT	41.67%	36.00%	19.58%	12.99%
SVM	38.75%	32.41%	28.33%	23.26%
NN	61.67%	62.26%	26.25%	21.00%
GBT	52.08%	48.28%	22.08%	15.89%
Best	61.67%	62.26%	35.00%	35.26%

have the same negative impact as brightness, with at most 6% change. We note that NN, GBT and K-NN models continue to yield the highest attack accuracy and F1 score despite changes in gamma.

Impact of ambient light conditions. Recall from Section 3.1.2 that our data collection was performed in the evening in a room illuminated by a ceiling lamp. In this section, we vary this ambient light condition by executing LuxTrack in two alternative conditions: (i) reduced light conditions in which the room is illuminated only by a distant bedside lamp, (ii) high ambient light conditions in which the room is illuminated by sunlight on a sunny day. The results of LuxTrack in these two conditions are reported in Table 4.6.

The results in Table 4.6 show that LuxTrack remains effective under reduced ambient light conditions, though with a slight decrease in accuracy and F1 score. Since limited illumination is provided by the bedside lamp in reduced light conditions, the ALS records very low values (close to 0) for activities that have dark backgrounds. Hence, such activities become difficult to differentiate from one another. On the other hand, under sunlight conditions, the accuracy of LuxTrack decreases substantially (∼35%). This result shows that sunlight dominates the ambient light conditions, reducing the effect of the illuminance caused by the laptop display. Therefore, the ALS readings no longer reflect the laptop activity under strong daylight. This causes our attack accuracy to drop. Finally, we observe from Table 4.6 that NN, GBT, and K-NN models continue to be the better-performing models. Interestingly, under daylight conditions, K-NN yields higher accuracy and F1 score compared to NN and GBT, which is typically not the case in the remaining experiments.

Impact of smartphone models. We used a Samsung Galaxy A72 smartphone in our

Table 4.7 Accuracy and F1 scores of LuxTrack using two alternative smartphones (Nokia 3.1 Plus and Oppo A5s).

	Nokia 3.1 Plus		**Oppo A5s**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	44.17%	37.50%	45.42%	42.65%
K-NN	74.58%	73.91%	64.17%	63.11%
DT	60.00%	57.01%	52.92%	51.53%
SVM	58.75%	55.86%	62.08%	63.23%
NN	78.33%	77.79%	80.83%	80.53%
GBT	77.50%	76.86%	70.42%	69.93%
Best	78.33%	77.79%	80.83%	80.53%

initial data collection. However, different smartphone models can be equipped with different ALSs, therefore we also test the effectiveness of LuxTrack using two other smartphone models: Nokia 3.1 Plus and Oppo A5s. We note that the ALSs on these phones have different manufacturers and technical specifications. For example, the ALS on Samsung Galaxy A72 is manufactured by Sensortek, its model is STK31610, and its resolution is 0.5 lux. In contrast, the ALS on Oppo A5s is manufactured by ams OSRAM, its model is TSL2540, and its resolution is 1.0 lux.

The results of LuxTrack using Nokia 3.1 Plus and Oppo A5s are shown in Table 4.7. We observe that the accuracy and F1 scores obtained using these two smartphones are very similar to those obtained using Samsung A72 (rightmost columns in Table 4.1). The attack is approximately 1% less accurate using Nokia 3.1 Plus compared to Samsung A72, whereas it is approximately 1% more accurate using Oppo A5s compared to Samsung A72. Many of the observations regarding individual ML models are also similar for different smartphones. Considering these observations, we can conclude that LuxTrack is not specific to Samsung A72; it is generally effective on different smartphone models. The fact that LuxTrack is effective on multiple smartphones equipped with different ALSs demonstrates the generalizability of our attack.

Impact of the spatial relationship between the ALS and the display (varying angles). Recall our hardware placement and the spatial relationship between the smartphone ALS and the laptop display from Section **??**. In this section, we explore the impact of rotating the smartphone away from the laptop screen. In particular, we tested two rotation settings: (i) 30 vertical rotation away from the laptop screen, and (ii) 45 horizontal rotation away from the laptop screen. The results are given in Table 4.8.

As the smartphone is rotated away from the laptop screen, it becomes harder for the

Table 4.8 Accuracy and F1 scores of LuxTrack under two rotation settings: 30 vertical rotation away from the laptop screen, 45 horizontal rotation away from the laptop screen.

	30 Vertical		**45 Horizontal**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	35.83%	31.89%	31.67%	30.00%
K-NN	50.83%	48.28%	37.92%	30.76%
DT	40.83%	36.79%	26.25%	21.08%
SVM	38.33%	35.78%	44.17%	39.81%
NN	66.67%	66.48%	62.08%	62.25%
GBT	57.92%	53.35%	60.83%	56.58%
Best	66.67%	66.48%	62.08%	62.25%

ALS to read the light emitted from the screen. Therefore, we would expect LuxTrack's accuracy to drop. The results in Table 4.8 confirm this intuition, both for horizontal and vertical rotations. According to the results, 30 vertical rotation causes LuxTrack's accuracy to drop by 13% and 45 horizontal rotation causes accuracy to drop by 17% compared to the default setting (no rotation). Higher the degree of rotation, higher the accuracy drop. Yet, we highlight that LuxTrack is still quite effective despite the substantially high degrees of rotation – its accuracy remains at 66% and 62% respectively, whereas the accuracy of a random guess is 12.5%. Hence, we conclude that while LuxTrack is indeed affected by the spatial relationship between the ALS and the laptop screen, its accuracy remains high despite substantial amounts of rotation.

4.1.5 Results with LOO-CV

Table 4.9 Comparison between LuxTrack, LightSpy and raw ALS when LOO-CV is used instead of 10-fold cross validation.

	Raw ALS		**LightSpy**		**LuxTrack**	
Model	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**
NB	33.08%	28.30%	55.10%	52.69%	55.67%	54.38%
K-NN	46.54%	42.81%	54.90%	54.73%	66.63%	65.82%
DT	31.15%	21.81%	56.35%	48.64%	60.48%	56.67%
SVM	38.56%	35.92%	46.90%	44.64%	63.85%	63.68%
NN	49.62%	48.53%	-	-	72.02%	71.94%
GBT	56.15%	55.38%	-	-	75.29%	75.38%
Best	56.15%	55.38%	56.35%	54.73%	75.29%	75.38%

In this section, we perform experiments with the Leave-One-Out Cross Validation (LOO-

CV) technique instead of 10-fold cross validation. To enable making comparisons between LOO-CV and 10-fold cross validation, we use the same dataset (same participants, activities, durations, etc.) as in Section 4.1.1 but retrain the models using LOO-CV rather than 10-fold cross validation. In LOO-CV, one human subject is iteratively picked for testing, and the remaining subjects are used for training. We repeat this process for each subject and then report the average accuracy and F1 scores for all iterations. LOO-CV enables us to explore how accurate our models are when they are tested with data from a previously unseen human subject.

The results of LOO-CV experiments are given in Table 4.9. In general, results obtained with LOO-CV are similar to results obtained with 10-fold cross validation (Table 4.1). In terms of the best-performing ML models, small increases in accuracy and F1 score (2-3%) are obtained under LOO-CV compared to 10-fold cross validation when models are trained with raw ALS data. In contrast, the accuracy and F1 scores of LightSpy and LuxTrack are approximately 4-5% lower under LOO-CV compared to 10-fold cross validation. When we study the individual ML models, we observe that NB and SVM perform better under LOO-CV whereas the remaining models perform better under 10-fold cross validation.

Overall, considering all results, we conclude that LOO-CV can slightly decrease attack accuracy and F1 score compared to 10-fold cross validation, but their results are similar. In addition, there are also cases in which LOO-CV yields higher attack accuracy and F1 score compared to 10-fold cross validation. Thus, the attack's performance is similar under LOO-CV and 10-fold cross validation.

4.1.6 Intra-Activity Diversity

Recall from Section 3.1.2 that for each activity (such as shopping on Amazon, reading PDFs on Adobe Acrobat, watching Youtube, etc.) we allow participants to freely choose which Amazon product to look at, which pdf document to read, which Youtube video to watch, and so forth. Hence, the data we collected contains a substantial amount of *intra-activity diversity*, i.e., different participants have diverse behaviors for each activity. In this section, we evaluate a "simpler" scenario in which there is no intra-activity diversity for the Youtube activity. That is, we ask each participant to watch the same Youtube video. Since this is an easier scenario, we expect LuxTrack to perform more accurately. To perform this experiment, we collected additional data using the same data collection procedures as in Section 3.1.2, using the same set of participants, hardware, data collection durations, and so forth. The key difference is that each participant is asked to watch the same Youtube video rather than different videos.

Table 4.10 Accuracy and F1 scores of LuxTrack when each participant is asked to watch the same Youtube video.

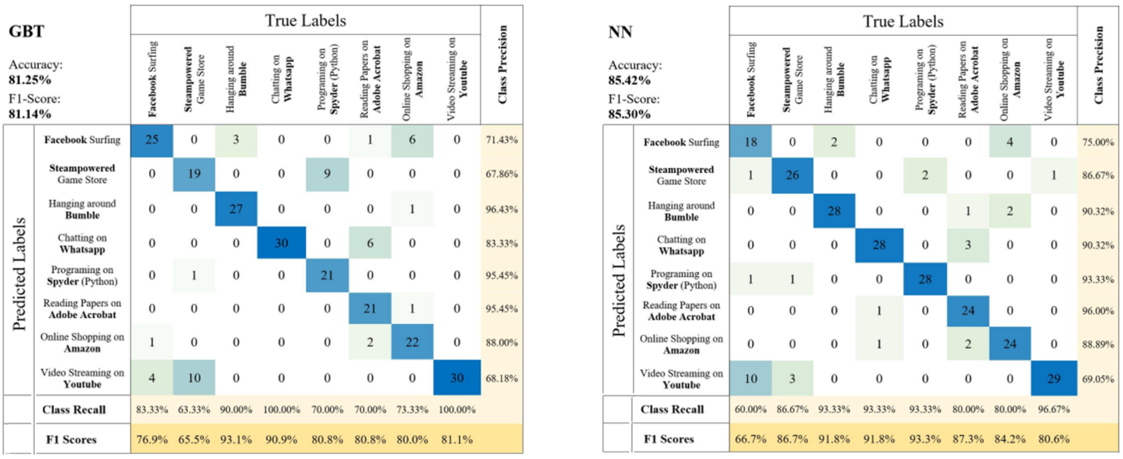
	LuxTrack (same video)	
Model	**Accuracy**	**F1 Score**
NB	52.92%	45.99%
K-NN	79.17%	78.26%
DT	67.92%	68.31%
SVM	62.92%	59.30%
NN	85.42%	85.30%
GBT	81.25%	81.14%
Best	85.42%	85.30%

The results of this scenario are shown in Table 4.10. Compared to Table 4.1, we observe that the accuracy and F1 scores of all models increase. NB, NN, and GBT models especially have large increases (around 5-6%) compared to K-NN and SVM (around 2%). NN continues to be the best-performing model, followed by GBT. Notably, the accuracy of LuxTrack increases from 79.58% to 85.42% in this simpler scenario.

Next, we analyze the reason behind this increase by studying the confusion matrices of the two best-performing models: NN and GBT. The confusion matrices are presented in Figure 4.2. When we compare these matrices with Figure 4.1, we observe that the recall values of the Youtube activity are substantially higher in Figure 4.2. The recall values of other activities are similar in both figures. In addition, there is a substantial increase in the precision of some activities such as Steampowered and Youtube. While the increase in the precision of Youtube activity is expected, the Steampowered activity is curious. Upon analyzing this behavior, we saw from Figure 4.1 that many true samples that belonged to the Youtube activity were frequently predicted incorrectly as belonging to the Steampowered activity by NN and GBT models. However, this did not happen when all subjects watched the same Youtube video (Figure 4.2). Hence, we can conclude that the accuracy increase observed in Table 4.10 is due to both easier prediction of the Youtube activity, and also better prediction of activities that are frequently confused with Youtube.

4.1.7 Extension to Multitasking

In the previous experiments, we assumed that laptop users are performing one activity at a time. However, it is also possible that users multitask with multiple applications displayed on the laptop screen at the same time. For example, users can divide their



GBT

Accuracy: **81.25%**

F1-Score: **81.14%**

Predicted Labels \ True Labels	Facebook Surfing	Steampowered Game Store	Hanging around Bumble	Chatting on Whatsapp	Programing on Spyder (Python)	Reading Papers on Adobe Acrobat	Online Shopping on Amazon	Video Streaming on Youtube	Class Precision
Facebook Surfing	25	0	3	0	0	1	6	0	71.43%
Steampowered Game Store	0	19	0	0	9	0	0	0	67.86%
Hanging around Bumble	0	0	27	0	0	0	1	0	96.43%
Chatting on Whatsapp	0	0	0	30	0	6	0	0	83.33%
Programing on Spyder (Python)	0	1	0	0	21	0	0	0	95.45%
Reading Papers on Adobe Acrobat	0	0	0	0	0	21	1	0	95.45%
Online Shopping on Amazon	1	0	0	0	0	2	22	0	88.00%
Video Streaming on Youtube	4	10	0	0	0	0	0	30	68.18%
Class Recall	83.33%	63.33%	90.00%	100.00%	70.00%	70.00%	73.33%	100.00%	
F1 Scores	76.9%	65.5%	93.1%	90.9%	80.8%	80.8%	80.0%	81.1%	

(a) GBT

NN

Accuracy: **85.42%**

F1-Score: **85.30%**

Predicted Labels \ True Labels	Facebook Surfing	Steampowered Game Store	Hanging around Bumble	Chatting on Whatsapp	Programing on Spyder (Python)	Reading Papers on Adobe Acrobat	Online Shopping on Amazon	Video Streaming on Youtube	Class Precision
Facebook Surfing	18	0	2	0	0	0	4	0	75.00%
Steampowered Game Store	1	26	0	0	2	0	0	1	86.67%
Hanging around Bumble	0	0	28	0	0	1	2	0	90.32%
Chatting on Whatsapp	0	0	0	28	0	3	0	0	90.32%
Programing on Spyder (Python)	1	1	0	0	28	0	0	0	93.33%
Reading Papers on Adobe Acrobat	0	0	0	1	0	24	0	0	96.00%
Online Shopping on Amazon	0	0	0	1	0	2	24	0	88.89%
Video Streaming on Youtube	10	3	0	0	0	0	0	29	69.05%
Class Recall	60.00%	86.67%	93.33%	93.33%	93.33%	80.00%	80.00%	96.67%	
F1 Scores	66.7%	86.7%	91.8%	91.8%	93.3%	87.3%	84.2%	80.6%	

(b) NN

Figure 4.2 Confusion matrices of GBT and NN models in the simpler scenario where all users watch the same Youtube video.

screen into two parts – they can read a pdf document in one half and write code in the other half. Then, an interesting aspect to consider is whether LuxTrack can be extended to multitasking users.

To evaluate LuxTrack on multitasking users, we constructed the following experiment. First, we chose 4 activities that could be multitasked by users in real life: *Reading pdf documents via Adobe Acrobat, Using Bumble, Watching Youtube videos, Programming on Spyder*. Then, we considered all pairwise combinations of these 4 activities, i.e., we have $\binom{4}{2} = 6$ combinations. For each combination, we collected data from participants who multitask with those combinations of activities on the laptop screen by dividing the screen into two halves. The same settings as in Section 3.1.2 were used in collecting this data (same set of participants, hardware, sample sizes, durations, etc.). Then, we used the same feature extraction and ML model training phases of LuxTrack, by treating each combination as one class. Thus, we obtained ML models for a 6-class classification problem.

The accuracy and F1 score results of our models are given in Table 4.11. Results show that LuxTrack is highly effective in multitasking scenarios as well, achieving higher than 87% accuracy. In addition to NN and GBT, which are typically the best-performing models in the previous experiments, we observe that other models also perform well. In fact, K-NN (which used to be the third-best model in previous experiments) is now the best-performing model in Table 4.11. The accuracy and F1 scores of the remaining models are also quite strong. Overall, the results indicate that LuxTrack can be successfully extended and applied to infer activities of multitasking users, in addition to users performing a

Table 4.11 Accuracy and F1 scores of LuxTrack when participants are multitasking with two activities.

	LuxTrack (Multitasking)	
Model	**Accuracy**	**F1 Score**
NB	81.67%	80.07%
K-NN	87.22%	87.18%
DT	86.67%	86.57%
SVM	83.89%	83.83%
NN	85.00%	84.97%
GBT	84.44%	84.12%
Best	87.22%	87.18%

single activity.

4.2 Subject Inference Attack Results and Discussions

The dataset, which was recorded from 24 different subjects while performing 19 activities, will be provided to the clients after removing the information related to the subject. Our attack scenario in this research is predicting the subject when the subject id of the test data received has been erased. To design such an attack system, we introduce two solutions to select portion of dataset and train the models on. Test results will provide us the solution with the best performance score. We evaluate the effectiveness of our attack system using the dataset we obtained. All results are reported with 4-fold cross validation. We use accuracy metric to evaluate the success rate of our attack. In this section, we test the performance of models proposed as two different approaches for subject prediction. In the first approach, we predict the subject directly from the entire dataset, and in the second approach, we first predict the activity of the data sample and then predict the subject who performed it in the subset of the data related to only the activity type of that sample. The results obtained from each approach are discussed in the following sub sections.

4.2.1 Results of the first cut solution

After training the subject prediction models on 4 trials recorded from every 19 activity types performed by each subject, we test them for only remaining trial from 5. The accuracy of the models is according to the table 4.12 where 75.44% is the highest one we have achieved to predict the subjects of trials.

To better understand which activity types contribute most to subject identifiability, we

Table 4.12 Accuracy of subject prediction models in first cut solution

Models	DT	GBT	XGBoost	Light GBM
Prediction Accuracy	54.82 %	70.61 %	74.56 %	75.44 %

analyze the partial accuracy of subject prediction models across individual activities. Table 4.13 presents the prediction performance for each activity type, detailing how accurately the models identify subjects based on the activity trial.

As seen in Table 4.13, trials of walking and jogging activities—regardless of being performed slowly or quickly—are generally predicted with low accuracy across all four classifiers. In contrast, six specific activity types consistently lead to high subject prediction accuracy: "Quickly sit in a low height chair, wait a moment, and up quickly", "Quickly/slowly sit in a half height chair, wait a moment, and up quickly/slowly", "Sitting a moment, lying quickly, wait a moment, and sit again", and "Standing, slowly bending at/without bending knees, and getting up". These activities appear to encode more distinctive personal movement signatures.

Beyond this general observation, several deeper insights emerge. First, activities involving low-height or half-height seated transitions result in the highest accuracies—up to 100%—indicating that such tasks may amplify subject-specific biomechanical patterns. Second, fall-like or instability-prone activities, such as "collapsing into a chair" or "stumbling while walking", also yield relatively strong performance, suggesting that individual strategies for maintaining or recovering balance are distinguishable. Finally, the consistent ranking of activities across all classifiers—whether high or low—demonstrates that the activity type itself is a major determinant of subject identifiability, independent of the model architecture.

4.2.2 Results of the Proposed Solution

Table 4.14 presents the accuracy of subject prediction models within the framework of our proposed two-stage attack approach. In this design, an activity classifier is first trained on the full dataset to identify the activity type of an unlabeled trial. Once the activity is predicted, the trial is passed to a second classifier trained only on the subset of data corresponding to that predicted activity to infer the subject identity.

As shown in Table 4.14, the highest subject prediction accuracy—89.04%—is achieved when the decision tree (DT) model is used for activity prediction, followed by the XGBoost model for subject prediction.

Table 4.13 Partial accuracy of subject prediction models for trials of each activity type. While walking and jogging trials—regardless of speed—tend to yield lower subject prediction accuracy, certain complex seated or transition activities consistently result in higher accuracy across all four models.

Activity Specific	Subject Prediction Models			
	DT	gbt	xgboost	lightgbm
Walking slowly	8.33%	25.00%	33.33%	33.33%
Walking quickly	16.67%	29.17%	45.83%	45.83%
Jogging slowly	4.17%	8.33%	20.83%	16.67%
Jogging quickly	8.33%	8.33%	12.50%	12.50%
Walking upstairs and downstairs slowly	50.00%	75.00%	58.33%	70.83%
Walking upstairs and downstairs quickly	66.67%	83.33%	95.83%	95.83%
Slowly sit in a half height chair, wait a moment, and up slowly	66.67%	91.67%	87.50%	87.50%
Quickly sit in a half height chair, wait a moment, and up quickly	75.00%	91.67%	95.83%	95.83%
Slowly sit in a low height chair, wait a moment, and up slowly	83.33%	91.67%	91.67%	95.83%
Quickly sit in a low height chair, wait a moment, and up quickly	70.83%	95.83%	100.00%	100.00%
Sitting a moment, trying to get up, and collapse into a chair	62.50%	75.00%	87.50%	91.67%
Sitting a moment, lying slowly, wait a moment, and sit again	45.83%	83.33%	87.50%	91.67%
Sitting a moment, lying quickly, wait a moment, and sit again	70.83%	87.50%	95.83%	95.83%
Being on one's back change to lateral position, wait a moment, and change to one's back	58.33%	70.83%	66.67%	75.00%
Standing, slowly bending at knees, and getting up	79.17%	87.50%	91.67%	91.67%
Standing, slowly bending without bending knees, and getting up	79.17%	100.00%	91.67%	91.67%
Standing, get into a car, remain seated and get out of the car	66.67%	75.00%	83.33%	75.00%
Stumble while walking	54.17%	75.00%	83.33%	79.17%
Gently jump without falling (trying to reach a high object)	75.00%	87.50%	87.50%	87.50%
Average Accuracy	54.82%	70.61%	74.56%	75.44%

Table 4.14 Accuracy of subject prediction models for models selected to already predict the activity in our proposed solution

		Subject Prediction			
	Models	DT	GBT	XGBoost	LightGBM
Activity Prediction	DT	65.79%	70.83%	89.04%	87.94%
	GBT	66.45%	69.74%	87.94%	86.40%
	XGBoost	65.79%	69.52%	87.50%	85.96%
	LightGBM	65.57%	69.08%	87.28%	85.53%

This combination outperforms all other model pairings. Notably, across all activity classifiers, the XGBoost and LightGBM models consistently yield high subject prediction accuracy in the second stage, suggesting their superior ability to capture subject-specific patterns within activity-specific subspaces. On the other hand, DT and GBT used in the second step consistently deliver lower accuracy, reinforcing the advantage of gradient-boosted frameworks in high-resolution classification tasks.

Table 4.15 Accuracy of activity prediction models in the 1st step of our proposed solution

Models	DT	GBT	XGBoost	Light GBM
Prediction Accuracy	83.55 %	88.60 %	91.45 %	93.42 %

Table 4.15 further details the performance of activity prediction models used in the first stage of the attack. As expected, LightGBM achieves the highest activity classification accuracy at 93.42%, followed closely by XGBoost at 91.45%. Interestingly, however, these models do not always yield the highest subject prediction performance when used as the first-stage classifier. This suggests that higher activity classification accuracy does not necessarily correlate with higher overall attack success—possibly due to the way each model's prediction influences the construction of activity-specific sub-datasets for subject inference. In particular, DT may provide more "conservative" activity predictions that result in cleaner sub-datasets, which in turn help XGBoost achieve higher subject prediction accuracy.

These results collectively demonstrate that the effectiveness of the subject inference attack depends not only on the classification accuracy of each stage but also on the interplay between the first and second stage classifiers. Thus, careful model pairing is essential to maximize the attack potential in practical scenarios.

In the second stage of the proposed approach, subject prediction is performed with the assumption that the attacker already knows the activity type of the trial—information that was inferred in the first stage. As a result, instead of training models on the entire dataset, we train 19 separate subject prediction models (one per activity) for each of the four classifiers. Each model is trained on activity-specific data that includes four out of five trials per activity for all 24 subjects. This enables classifiers to specialize in capturing subject-specific patterns within a particular activity context.

Table 4.16 presents the accuracy of these subject prediction models across all activity types. The most striking observation is the significantly improved performance compared to earlier attack stages. In particular, activities such as “jogging slowly,” “walking quickly,” and “walking slowly” achieve perfect subject identification accuracy (100%) with XGBoost and LightGBM, and near-perfect results with GBT and DT. Even “jogging quickly,” which was previously associated with low prediction accuracy in Table 4.13, now yields perfect accuracy with XGBoost and LightGBM and 87.50% with GBT—demonstrating that activity-specific tailoring of models drastically enhances subject discrimination.

These results suggest that when activities are known, the intra-activity variability in how individuals perform them becomes a strong biometric signal. In essence, the “noise” introduced by inter-activity variation is eliminated, allowing the classifier to focus solely on person-specific motion traits. Moreover, the four walking and jogging activities, which performed poorly in the generalized setting, now become some of the best-performing categories, indicating that fine-grained gait differences are quite unique when observed within a single activity frame.

Conversely, several seated or transition activities that previously produced high accuracy under the global model setup now exhibit relatively lower performance when isolated—such as “sitting a moment, lying slowly, wait a moment, and sit again” and “being on one’s back change to lateral position. . . ”. This shift highlights the possible effect of reduced data diversity in individual activity subsets or the increased homogeneity of movement in such tasks.

Overall, these findings underline the strength of the proposed two-step approach, where decoupling activity and subject classification not only improves performance but also reveals hidden structure in subject motion patterns—structure that becomes more apparent when activity context is fixed. They also confirm the robustness of XGBoost and LightGBM as subject identifiers across a wide range of activity types, outperforming DT and GBT in most scenarios.

Table 4.16 Accuracy of subject prediction models on datasets of already predicted activities in the 1st step of our proposed solution

Activities	Subject Prediction Models			
	DT	gbt	xgboost	lightgbm
Walking slowly	66.67%	95.83%	100.00%	100.00%
Walking quickly	75.00%	91.67%	100.00%	100.00%
Jogging slowly	91.67%	100.00%	100.00%	100.00%
Jogging quickly	62.50%	87.50%	100.00%	100.00%
Walking upstairs and downstairs slowly	62.50%	70.83%	91.67%	91.67%
Walking upstairs and downstairs quickly	62.50%	70.83%	100.00%	91.67%
Slowly sit in a half height chair, wait a moment, and up slowly	70.83%	70.83%	87.50%	95.83%
Quickly sit in a half height chair, wait a moment, and up quickly	83.33%	79.17%	91.67%	91.67%
Slowly sit in a low height chair, wait a moment, and up slowly	66.67%	75.00%	91.67%	91.67%
Quickly sit in a low height chair, wait a moment, and up quickly	66.67%	66.67%	100.00%	91.67%
Sitting a moment, trying to get up, and collapse into a chair	58.33%	62.50%	83.33%	70.83%
Sitting a moment, lying slowly, wait a moment, and sit again	50.00%	33.33%	54.17%	54.17%
Sitting a moment, lying quickly, wait a moment, and sit again	54.17%	50.00%	83.33%	83.33%
Being on one's back change to lateral position, wait a moment, and change to one's back	54.17%	50.00%	66.67%	62.50%
Standing, slowly bending at knees, and getting up	70.83%	50.00%	75.00%	75.00%
Standing, slowly bending without bending knees, and getting up	83.33%	79.17%	91.67%	87.50%
Standing, get into a car, remain seated and get out of the car	58.33%	66.67%	66.67%	75.00%
Stumble while walking	79.17%	62.50%	95.83%	91.67%
Gently jump without falling (trying to reach a high object)	58.33%	62.50%	91.67%	83.33%

5. POTENTIAL COUNTERMEASURES AND THEIR IMPACT

This chapter presents the defense mechanisms proposed in response to the two distinct inference attacks explored in this thesis. The first set of countermeasures targets the ALS-based activity inference attack introduced in Section 3.1, where we demonstrated that ambient light sensor data can be exploited to accurately infer user activities. The second set focuses on mitigating the subject inference attack outlined in Section 3.2, which revealed privacy vulnerabilities in a fall-detection dataset originally designed for activity recognition. Each section introduces tailored defense strategies—ranging from signal obfuscation to noise injection at both feature and sensor levels—designed to reduce inference accuracy without significantly impairing legitimate functionality. We also analyze the trade-offs involved, aiming to strike a balance between privacy protection and task utility in real-world scenarios.

5.1 Countermeasures against ALS-based Activity Inference Attack

The high accuracy achieved by LuxTrack motivates the need for countermeasures which can reduce attack accuracy, i.e., make it difficult for an adversary to infer user's activity from ALS readings. At the same time, the countermeasure should not disable the smartphone OS or third-party app from using the ALS for legitimate reasons such as adjusting screen brightness or background color. The W3C Working Draft on ALS (World Wide Web Consortium, World Wide Web Consortium) describes two intuitive security principles for ALS, which can guide the design of countermeasures: (i) discard new ALS sensor readings if they fail to exhibit sufficient difference from previous readings, (ii) reduce the accuracy or precision of ALS readings. Since the first approach may interfere with the legitimate ALS sensor requests made by the OS or apps, we design countermeasures which follow the second principle, i.e., reducing accuracy or precision. Our countermeasures do not require any changes to the requests or responses of the smartphone OS or apps; therefore, they can be easily integrated with existing infrastructure.

We propose three potential countermeasures for LuxTrack: *Binning*, *Smoothing*, and

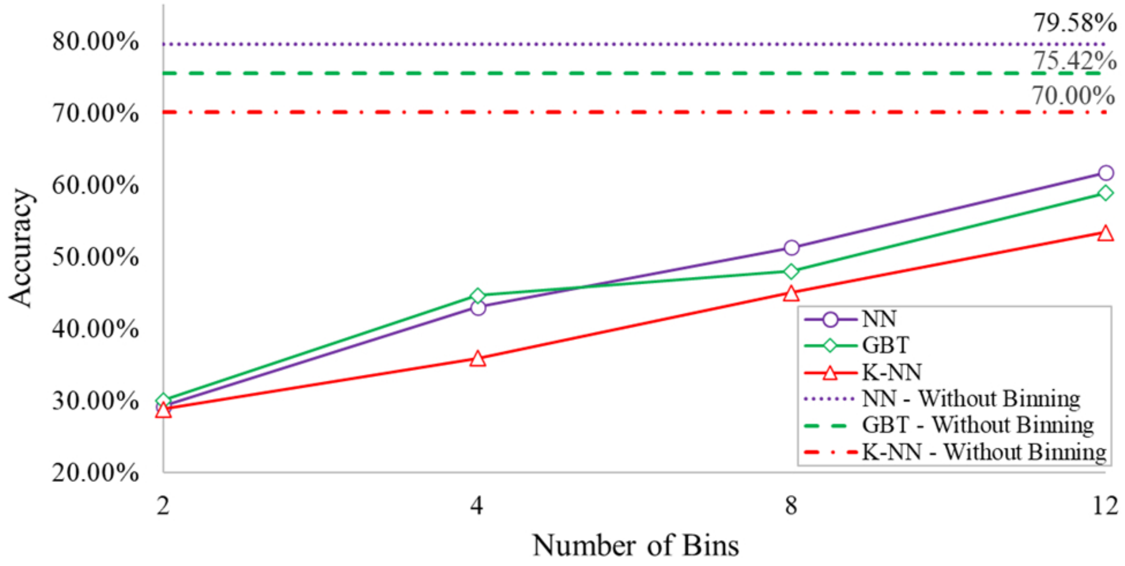


Figure 5.1 Attack accuracy of three ML models with highest original accuracy (NN, GBT and K-NN) on binned data, with varying number of bins n.

Noise Addition. These countermeasures rely on the same principle: after an ALS reading is obtained from the sensor (in units of lux), it is manipulated by the firmware or a trusted component of the OS before being passed on to untrusted components, e.g., a third-party smartphone app which may run an inference attack. Thus, the attacker is allowed to observe only the manipulated versions of the ALS readings. In addition to presenting the three countermeasures, we experimentally show that they are effective by demonstrating the decrease in attack accuracy achieved after each countermeasure is applied.

5.1.1 Binning

Our first countermeasure is binning, which discretizes ALS readings into n equal-sized bins. Let min and max denote the minimum and maximum possible ALS reading. n bins are created, each with size $s = (max - min)/n$. Then, for each x_i ($1 \leq i \leq |x|$), the value of x_i is replaced by its bin: $x_i \leftarrow x_i//s$ where // denotes integer division.

We tested this countermeasure using different numbers of bins: $n = 2, 4, 8, 12$. The results are shown in Figure 5.1. We compare the cases with and without binning for three different ML models: NN, GBT and K-NN. These three models were chosen because they originally had the highest attack accuracy (see Table 4.1). We observe that as we reduce the number of bins, attack accuracy decreases. This is because fewer bins imply higher data loss (bin sizes s are larger). For few bins such as $n = 2$ or 4, attack accuracy is as low as 30-45%, which implies strong resistance against LuxTrack attack. As n increases, attack accuracy recovers and approaches the original values for all three ML models.

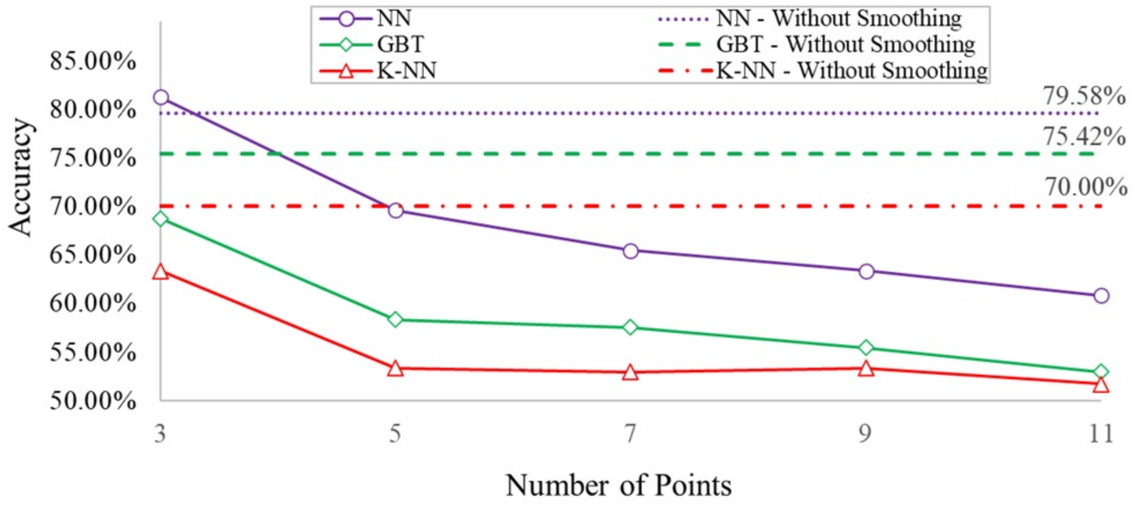


Figure 5.2 Attack accuracy of three ML models (NN, GBT and K-NN) after smoothing, with a varying number of readings (data points) k used in smoothing.

5.1.2 Smoothing

Our second countermeasure is smoothing, which smooths each ALS reading using its nearest neighbors, i.e., readings that are adjacent to this reading in the time series. More formally, x_i is replaced by:

$$x_i \leftarrow \frac{x_{i-\frac{k-1}{2}} + .. + x_i + .. + x_{i+\frac{k-1}{2}}}{k}$$

That is, each x_i is replaced by the average of $(k-1)/2$ points before it, $(k-1)/2$ points after it, and itself. Here, we denote by parameter k the number of ALS readings (number of data points) that will be used in smoothing. The higher the value of k, the smoother the resulting ALS time series will be after all x_i are replaced.

We tested this countermeasure using different $k = 3, 5, 7, 9, 11$. The results are shown in Figure 5.2. We use the same ML models as before (NN, GBT, K-NN) because they had the highest accuracy originally. We observe that as k increases, attack accuracy decreases. This is because an increase in k causes change-related features to be damaged. For example, SFC features such as variance, standard deviation, skewness, Variation Coefficient, as well as many features from the CFC, VFFC, and EFC categories are directly impacted by smoothing. Recalling the importance of these feature categories from our feature importance analysis, we can indeed expect a substantial decrease in attack accuracy when smoothing is used. On the other hand, it should be noted that even with $k = 9$ or 11 attack accuracy remains above 50%, which implies that other features which are less impacted by smoothing can still enable the attacker to achieve

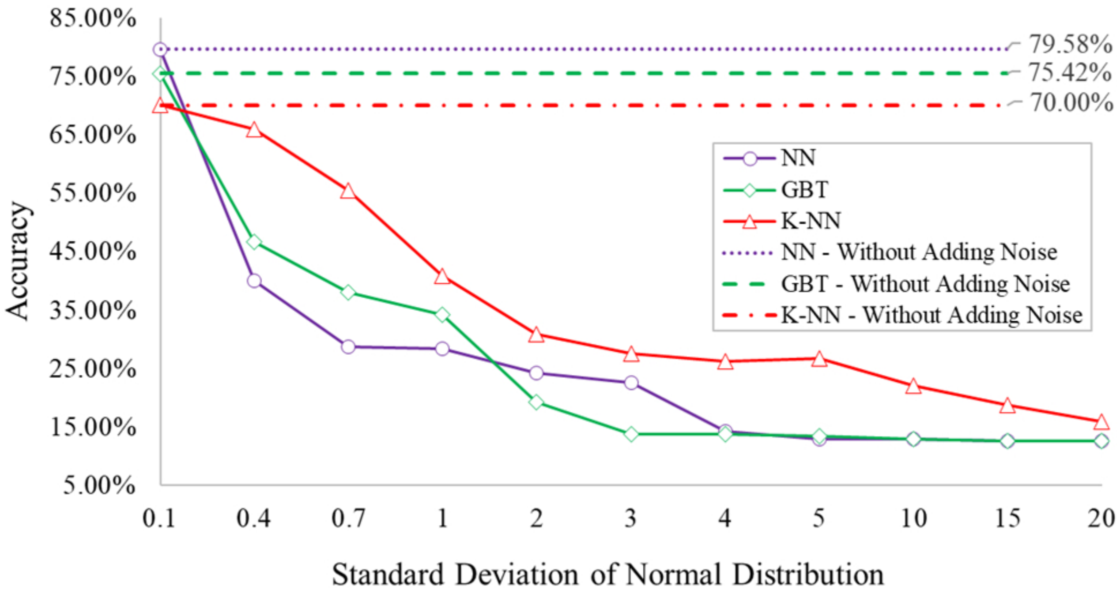


Figure 5.3 Attack accuracy of three ML models (NN, GBT and K-NN) after noise addition, with varying standard deviation σ used in the generation of noise samples.

semi-accurate attacks.

5.1.3 Noise Addition

Our third countermeasure is noise addition, which adds randomly and independently sampled Gaussian noise to each ALS reading. That is, x_i is replaced by:

$$x_i \leftarrow x_i + \mathcal{G}(0, \sigma)$$

where $\mathcal{G}(0, \sigma)$ denotes a Gaussian distribution (Normal distribution) with mean 0 and standard deviation σ. Typically, higher σ causes higher noise to be added to x_i, therefore ALS time series become more distorted.

We tested this countermeasure with σ values ranging from 0.1 to 20. The results are shown in Figure 5.3. When σ is very low such as $\sigma = 0.1$, the noise added is negligible, therefore there is very little impact on attack accuracy. As σ increases, the noise amount increases, therefore attack accuracy decreases. Attack accuracies of all three ML models are almost halved when $\sigma \geq 1$. With even higher σ, we observe that the accuracies of GBT and NN models drop as low as 12.5%, which is equal to the accuracy of a random guess (since we consider 8 activities in our dataset). This shows that if high amounts of noise are added to ALS readings, the LuxTrack attack can indeed become impossible to conduct.

5.1.4 Trade-offs Between Attack Accuracy and Legitimate Task Accuracy

In the experiments conducted thus far, we showed that our countermeasures can be effective against LuxTrack. However, each countermeasure has an internal parameter: number of bins n in binning, number of points k in smoothing, and standard deviation σ in noise addition. Countermeasures' effectiveness in preventing the attack depends on the values of these parameters. On the other hand, since the countermeasures manipulate ALS readings, legitimate apps which use the ALS readings for legitimate tasks are also affected. This creates a trade-off: Significant manipulation of ALS readings yields better protection against the attack but also reduced accuracy in legitimate tasks, whereas little manipulation of ALS readings maintains accuracy in legitimate tasks but offers little protection from attacks. A desirable goal is to achieve high attack protection with little accuracy reduction in legitimate tasks.

Towards this goal, in this section, we experimentally study the trade-offs between attack accuracy and legitimate task accuracy in order to guide the selection of the appropriate countermeasure and parameter. To measure attack accuracy, we use the same strategy as in the previous subsections: After we apply the countermeasure, we feed the resulting data to the LuxTrack pipeline (feature extraction and ML model) and obtain the attack accuracy result. To measure how well legitimate task accuracy is preserved, we consider a typical use case of the ALS sensor: changing the background color. Using ALS, the background color of a smartphone screen is adjusted by apps (such as Google Maps) from dark to light in bright environments and from light to dark in darker environments. Using Google Maps as a popular and representative mobile app, we experimentally found that the background color is changed from light to dark when illumination is below 7 lux. Thus, the screen should be set to dark when ambient light is < 7 lux and it should be set to light when ambient light is ≥ 7 lux. Accuracy in the legitimate task is measured as the percentage of cases in which the background color adjustment is correct according to the true ambient light of the environment.

Impact on Legitimate Task: Before demonstrating the trade-offs between attack accuracy and legitimate task accuracy, we first focus solely on the impact of countermeasures on the legitimate task (adjustment of screen background color). In this context, we define:

- True Positives: cases where screen background should be dark and it is correctly set to dark
- True Negatives: cases where screen background should be light and it is correctly set to light

Table 5.1 Impact of binning on legitimate task performance.

Num Bins (n)	Recall (TPR)	Selectivity (TNR)	Precision (PPV)	Negative Predictive Value (NPV)	Accuracy (ACC)
2	0	1	0	0.9822	0.9822
4	1	0.6815	0.054	1	0.6872
8	0.5817	1	1	0.9925	0.9925
12	1	0.9645	0.3384	1	0.9651

Table 5.2 Impact of smoothing on legitimate task performance.

Num Pts (k)	Recall (TPR)	Selectivity (TNR)	Precision (PPV)	Negative Predictive Value (NPV)	Accuracy (ACC)
3	0.9747	1	1	0.9995	0.9995
5	0.9397	0.9997	0.9817	0.9989	0.9986
7	0.8852	0.9996	0.9764	0.9979	0.9976
9	0.8405	0.9995	0.9686	0.9971	0.9967
11	0.8074	0.9993	0.9540	0.9965	0.9959

- False Positives: cases where screen background should be light but it is incorrectly set to dark

- False Negatives: cases where screen background should be dark but it is incorrectly set to light

Then, we measure recall (TPR), selectivity (TNR), precision (PPV), negative predictive value (NPV), and accuracy (ACC) according to their standard definitions. The results are shown in Tables 5.1, 5.2 and 5.3 for binning, smoothing and noise addition countermeasures, respectively.

From Table 5.1, we observe that binning has a substantial negative impact on legitimate task performance for all n values. When $n = 2$, TPR and PPV are equal to zero. When n $= 4$, PPV is close to zero. $n = 8$ or 12 are more favorable than $n = 2$ or 4, but even then, the TPR and PPV values are significantly low. As a result, we conclude that binning is not a desirable countermeasure to preserve legitimate task performance in general. From Table 5.2, we observe that smoothing can maintain high TNR, PPV, NPV and ACC for $k = 3$ to 11. Its main negative impact on legitimate task performance is the reduction in TPR. TPR is above 0.97 when $k = 3$ but it drops consistently as k is increased from 3 to

Table 5.3 Impact of noise addition on legitimate task performance.

σ	Recall (TPR)	Selectivity (TNR)	Precision (PPV)	Negative Predictive Value (NPV)	Accuracy (ACC)
0.1	1	1	1	1	1
0.4	0.9572	0.9984	0.9162	0.9992	0.9977
0.7	0.8677	0.9958	0.788	0.9976	0.9935
1	0.8638	0.9934	0.7025	0.9975	0.991
2	0.7374	0.9818	0.4244	0.9952	0.9775
3	0.7101	0.9599	0.2437	0.9945	0.9555
4	0.6051	0.9420	0.1593	0.9924	0.936
5	0.6401	0.9262	0.1362	0.9930	0.9211
10	0.5506	0.8645	0.0687	0.9906	0.8589
15	0.5272	0.8232	0.0514	0.9897	0.8179
20	0.5117	0.7779	0.0402	0.9887	0.7731

11. From Table 5.3, we observe that noise addition starts with perfect TPR, TNR, PPV, NPV and ACC when $\sigma = 0.1$. For such low σ, the added noise amounts are so low that there is no negative impact on legitimate task performance. However, as σ increases, all of TPR, TNR, PPV, NPV and ACC steadily decrease. The rate of decrease is fastest for TPR and PPV, whereas it is slower for others. This result shows that the legitimate task performance of noise addition is heavily dependent on the value of σ.

Analysis of Trade-offs: In Figures 5.4, 5.5 and 5.6, we provide graphs to compare the legitimate task accuracy and LuxTrack attack accuracy using three ML models with the highest attack accuracy (K-NN, NN, GBT). Figure 5.4 shows that binning is moderately effective in reducing attack accuracy when the number of bins $n = 8$ or 12, but for the attack to become less feasible, n should be as low as 2 or 4. However, for $n = 2$ or 4, we had observed from Table 5.1 that TPR, TNR, PPV and ACC of binning can be extremely low. Thus, when we consider the trade-off between attack accuracy and legitimate task performance, binning is not the most desirable countermeasure because a high decrease in legitimate task performance must be incurred for binning to become effective against LuxTrack.

When we analyze the trade-offs of smoothing by combining Figure 5.5 with Table 5.2, we observe that smoothing is generally good at maintaining high legitimate task performance across many k values. On the other hand, Figure 5.5 shows that attack accuracy also remains quite high (above 50%) for all k values. Thus, we conclude that while smoothing

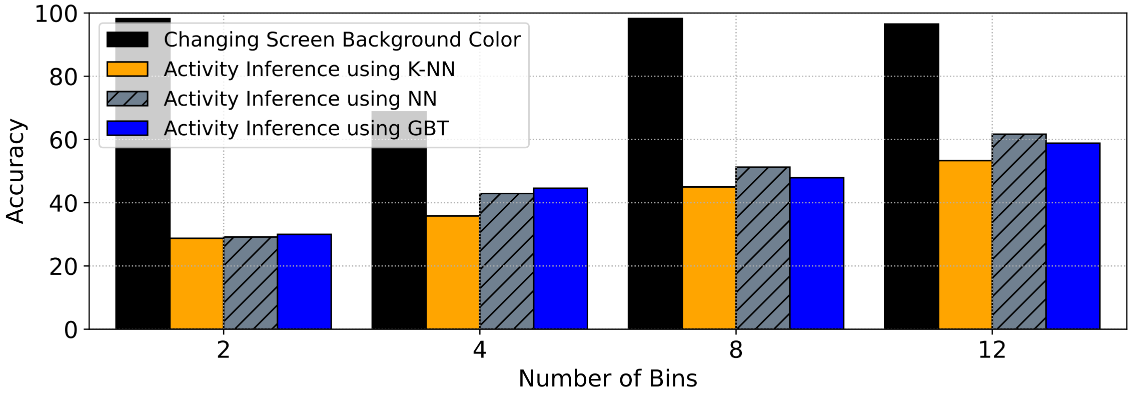


Figure 5.4 Trade-off between legitimate task accuracy (changing screen background color) and activity inference attack, for the binning countermeasure.

performs well in maintaining legitimate task performance, if high protection against the LuxTrack attack is desired, then it would not be the preferred choice of countermeasure. On the other hand, if the priority is to maintain as high legitimate task performance as possible while achieving *some* protection from the attack, then smoothing would be a suitable choice.

Finally, we study the trade-offs of the noise addition countermeasure using Figure 5.6 and Table 5.3. The protection offered by this countermeasure is not much when $\sigma = 0.1$ or 0.4. In contrast, near-perfect protection can be offered when $\sigma = 10$ or above, but this comes at the cost of substantial loss in legitimate task accuracy. $\sigma = 1$ and 2 seem to offer a good trade-off between legitimate task accuracy and protection from the attack. When $\sigma = 1$, attack accuracy is below 40% for all three ML models; in addition, legitimate task accuracy and other metrics in Table 5.3 remain high. When $\sigma = 2$, attack accuracy is below 30% for all three ML models; in addition, legitimate task accuracy remains above 97%. Thus, noise addition defense with σ close to 1 or 2 can be recommended to achieve a good trade-off.

5.1.5 Evaluation of Countermeasures on LightSpy

In addition to showing that our countermeasures are effective against LuxTrack, we also evaluate them on LightSpy to demonstrate their generality. Figure 5.7 contains the results for binning, Figure 5.8 contains the results for smoothing, and Figure 5.9 contains the results for noise addition. Overall, we observe that the countermeasures are effective against LightSpy as well, and their behaviors are similar to how they behave on LuxTrack. For example, with few numbers of bins such as 2 or 4, attack accuracy is as low as 20-45%. As the number of bins increases, attack accuracy also increases. Even with a relatively large number of bins ($n = 12$), there are significant drops in the attack accuracy of K-

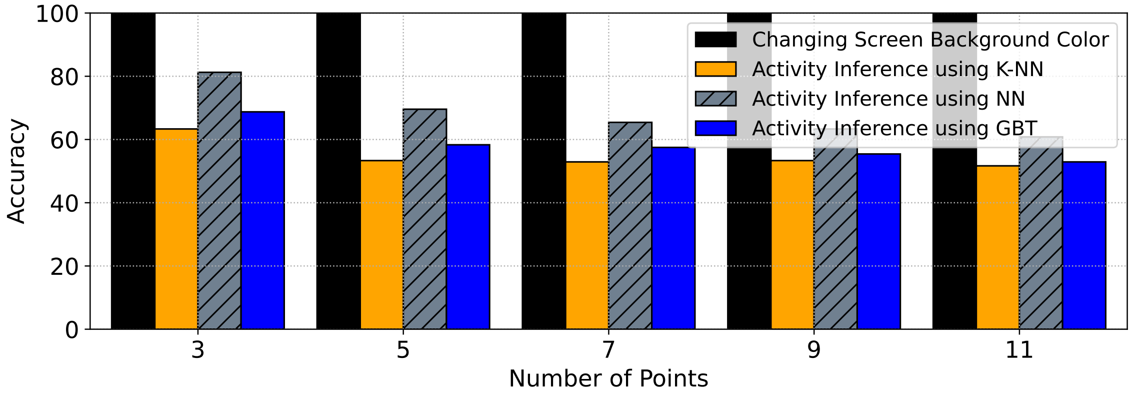


Figure 5.5 Trade-off between legitimate task accuracy (changing screen background color) and activity inference attack, for the smoothing countermeasure.

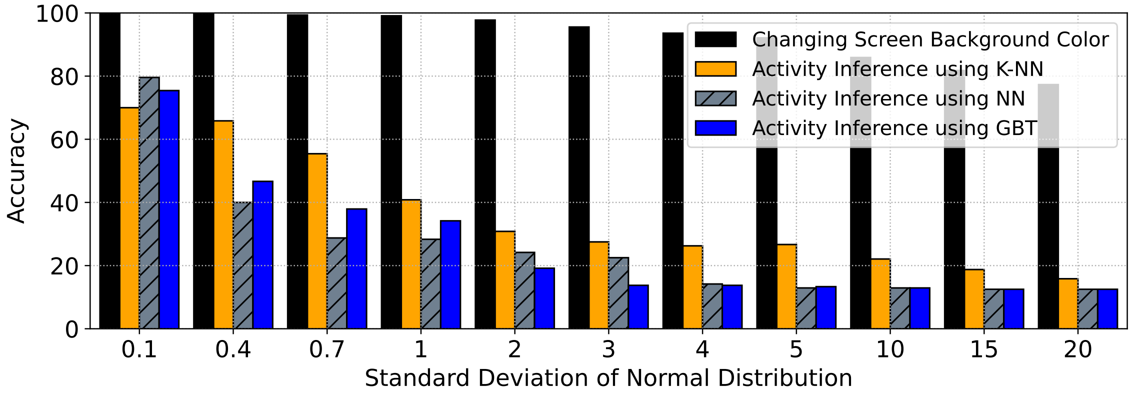


Figure 5.6 Trade-off between legitimate task accuracy (changing screen background color) and activity inference attack, for the noise addition countermeasure.

NN and SVM models. Also similar to LuxTrack, we observe from Figure 5.8 that as we increase the number of points k used in smoothing, attack accuracy decreases. The reduction in accuracy is small when $k = 3$ or 5; but substantial accuracy reductions are observed when $k \geq 7$. For the noise addition defense, we observe from Figure 5.9 that attack accuracy decreases quickly as the standard deviation σ of the noise exceeds 0.4. Similar to LuxTrack, attack accuracies are almost halved when $\sigma = 1$, and they approach the accuracy of a random guess when $\sigma \geq 4$.

5.2 Countermeasures against Subject Inference Attack

The foundation of our proposed countermeasure against subject-inference attacks is the deliberate manipulation of sensor data before it is transmitted to the client. The goal is to ensure minimal impact on the legitimate utility of the data—namely, activity prediction accuracy—while significantly reducing the accuracy of models developed by attackers

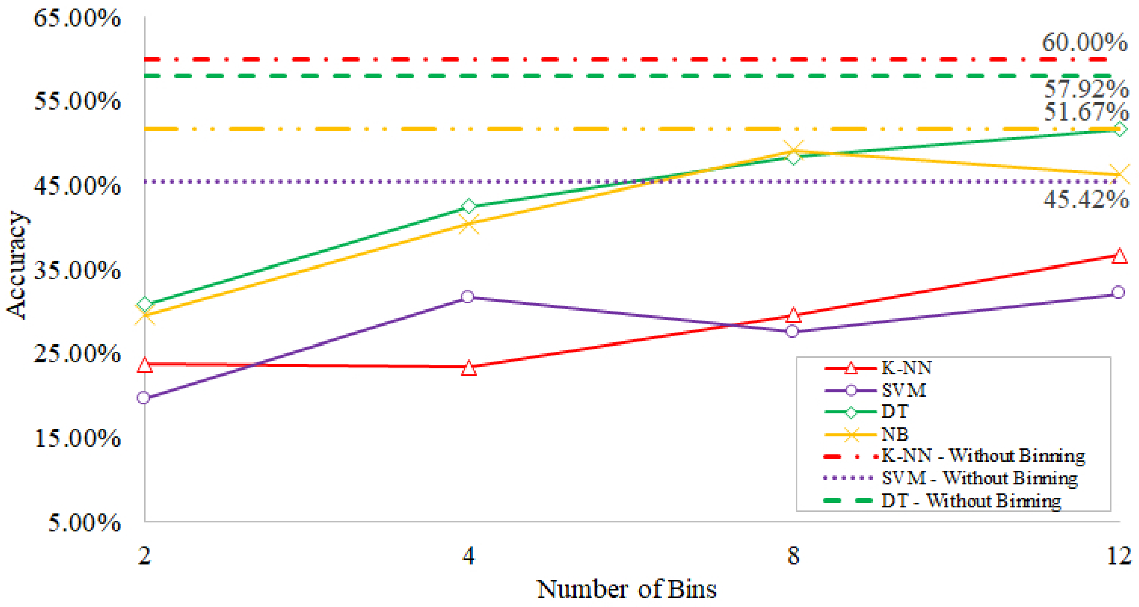


Figure 5.7 Attack accuracy of LightSpy models after our binning countermeasure is applied.

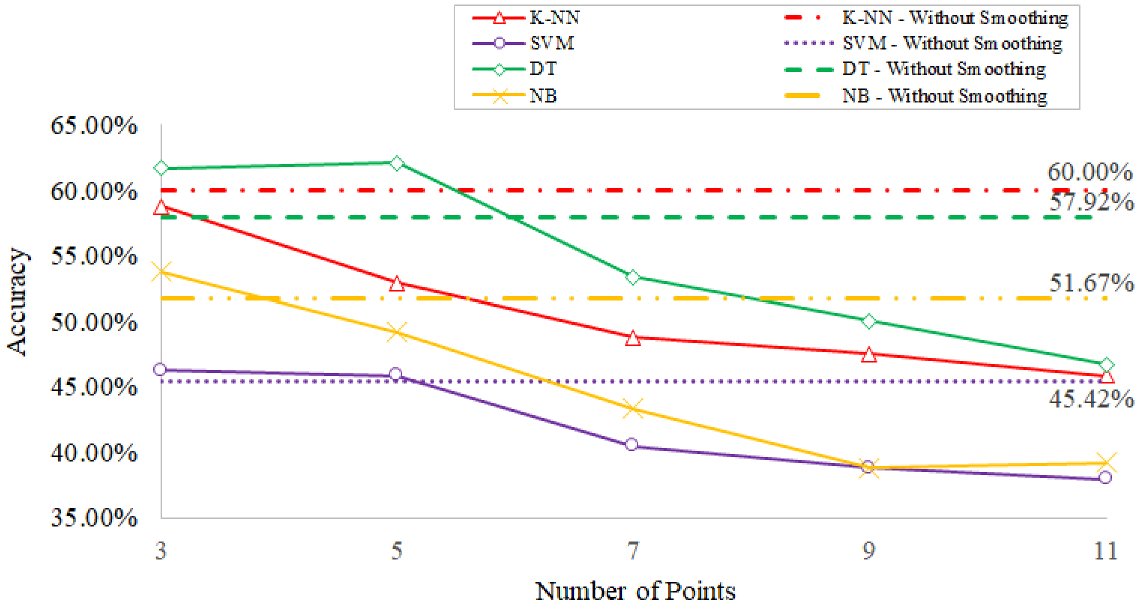


Figure 5.8 Attack accuracy of LightSpy models after our smoothing countermeasure is applied.

to infer subjects. To achieve this, we present example scenarios of defense mechanisms designed to counter specific types of attacks. These mechanisms can primarily be categorized into two groups: (1) those that manipulate features extracted from raw sensor data and (2) those that directly manipulate the raw sensor data itself. At the end of each scenario, depending on the defense type, either noisy features or raw data are sent

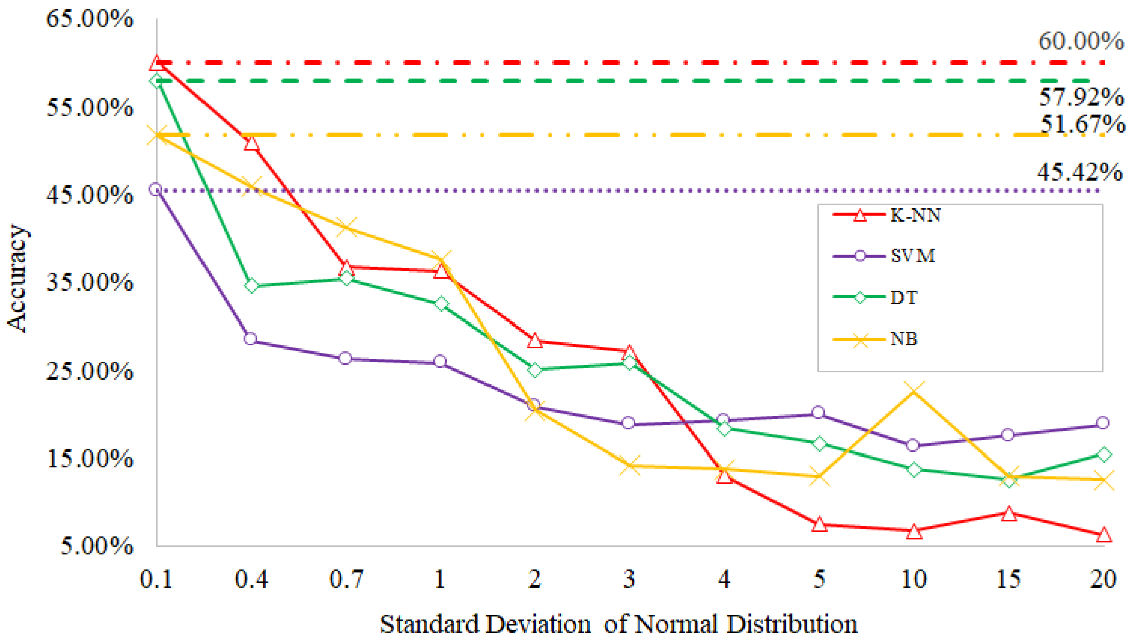


Figure 5.9 Attack accuracy of LightSpy models after our noise addition countermeasure is applied.

to the client. The attacker, with access to this data, applies four machine learning models (previously trained as a "first cut solution") to predict activities and subjects from the records. Below, we describe each defense mechanism in detail. The assumptions regarding the information available to each mechanism are summarized in the table 5.4.

5.2.1 Features-Based Defense Mechanisms

In the first three scenarios, the assumption is that the defense mechanism has access to features extracted from the raw sensor data. Instead of transmitting raw data from nine sensors, 156 significant features for subject classification and 156 significant features for activity classification are extracted and sent to the client. Given that 17 features overlap between the two sets, a total of 295 features are provided to the client in place of raw sensor data. These feature sets were selected based on lower p-values.

5.2.1.1 Random Feature Noise Injection Defense Mechanism (RFNI)

In this scenario, the defense system has access to all 295 features without knowing their specific importance for subject or activity classification. That is, the defense mechanism does not know which feature belongs to which set of 156 features. The defender randomly applies Gaussian noise with varying standard deviations to each feature. The noisy features are then sent to the client. The attacker, with malicious intent, uses the noisy features as input to its predictive models in an attempt to infer the subject.

Table 5.4 Summary of assumptions and information availability for each proposed defense mechanism.

Defender	**F/R**	Sig. Feat. (Subj.)	Sig. Feat. (Act.)	Informed of Raw Sensors' Sig.	# Noisy Sensor-Axes	# Noisy Features	ML Models Access & Feat. Provided	Noise Calc.	Search Method
RFNI	F	✗	✗	✗	0	295, 221, 147, 73	✗	✗	✗
SFNI	F	✓	✗	✗	0	156	✗	✗	✗
SOFNI	F	✓	✓	✗	0	139	✗	✗	✗
Uni-SANI	R	✗	✗	✗	6	0	✗	✗	✗
OptUni-SANI	R	✗	✗	✗	6	0	✓	✓	Binary Search
MinNoise	R	✗	✗	✓	2	0	✓	✓	Bayesian Search
AcConstrain	R	✗	✗	✓	2	0	✓	✓	Bayesian Search

The accuracy of the four predictive models for both activity and subject classification, based on the noisy data, is illustrated in Figures 5.10a and 5.10b. As shown, increasing the noise's standard deviation reduces the accuracy of all four models for predicting both the 19 activities and the 24 subjects.

This behavior can be attributed to the effect of Gaussian noise disrupting the dataś structure, making it more challenging for predictive models to identify meaningful patterns. However, since the defense mechanism does not distinguish between features critical for subject and activity classification, the noise is applied uniformly across all features. Consequently, the noise affects both categories of features indiscriminately.

Figure 5.10c compares the change in accuracy between activity prediction models and subject prediction models. Although increasing noise achieves the goal of reducing the accuracy of subject prediction, it significantly impacts activity prediction accuracy, which is undesirable. This result occurs because the overlapping features and shared information between subject and activity classification mean that noise affecting subject-relevant features also influences activity-relevant features. Additionally, the indiscriminate application of noise to all features results in degradation in model performance for both tasks, especially as the standard deviation of the noise increases.

Feature Subset Selection and Its Impact: In some cases, selecting a subset of features can yield better results for the models. Therefore, this scenario is extended by randomly selecting a specific percentage of features (e.g., 75%, 50%, and 25%) as input to the attacker's predictive models. To ensure comparability with previous results, the same noise levels are applied to the selected features. Figures 5.11, 5.12, and 5.13 illustrate the changes in accuracy for subject and activity prediction models, both as a function of noise standard deviation and relative to each other.

The results reveal that reducing the percentage of features available to the attacker produces distinct effects on subject and activity prediction models:

A. Subject Prediction Accuracy: Selecting fewer features increases the relative proportion of subject-relevant features in the subset, even when selected randomly. Subject classification models often rely on a smaller, distinct set of features, and this random selection process may retain a sufficient amount of this information. Furthermore, the addition of noise affects fewer features, reducing the overall disruption to subject-relevant patterns. As a result, subject prediction accuracy remains stable or even improves slightly in some cases.

B. Activity Prediction Accuracy: Activity classification relies on a broader range of

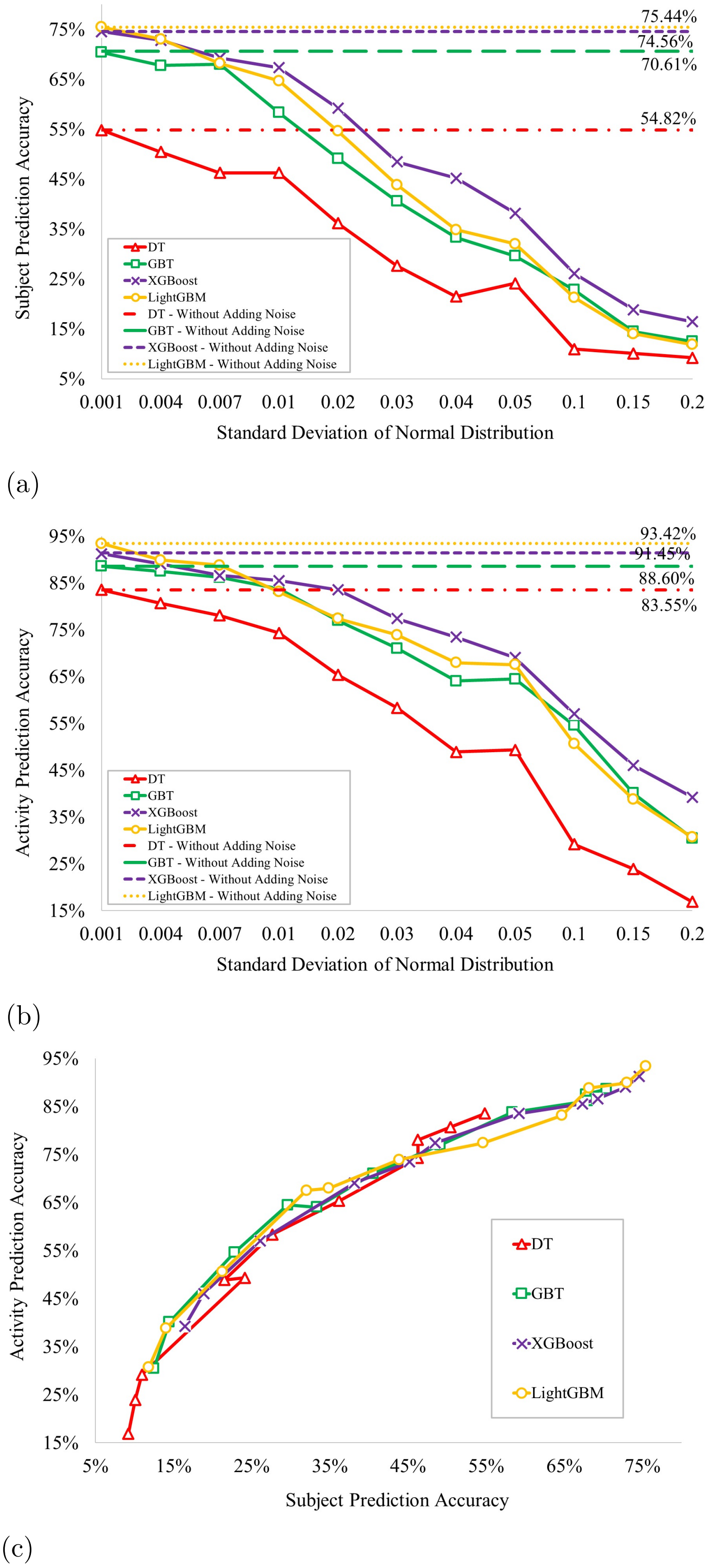
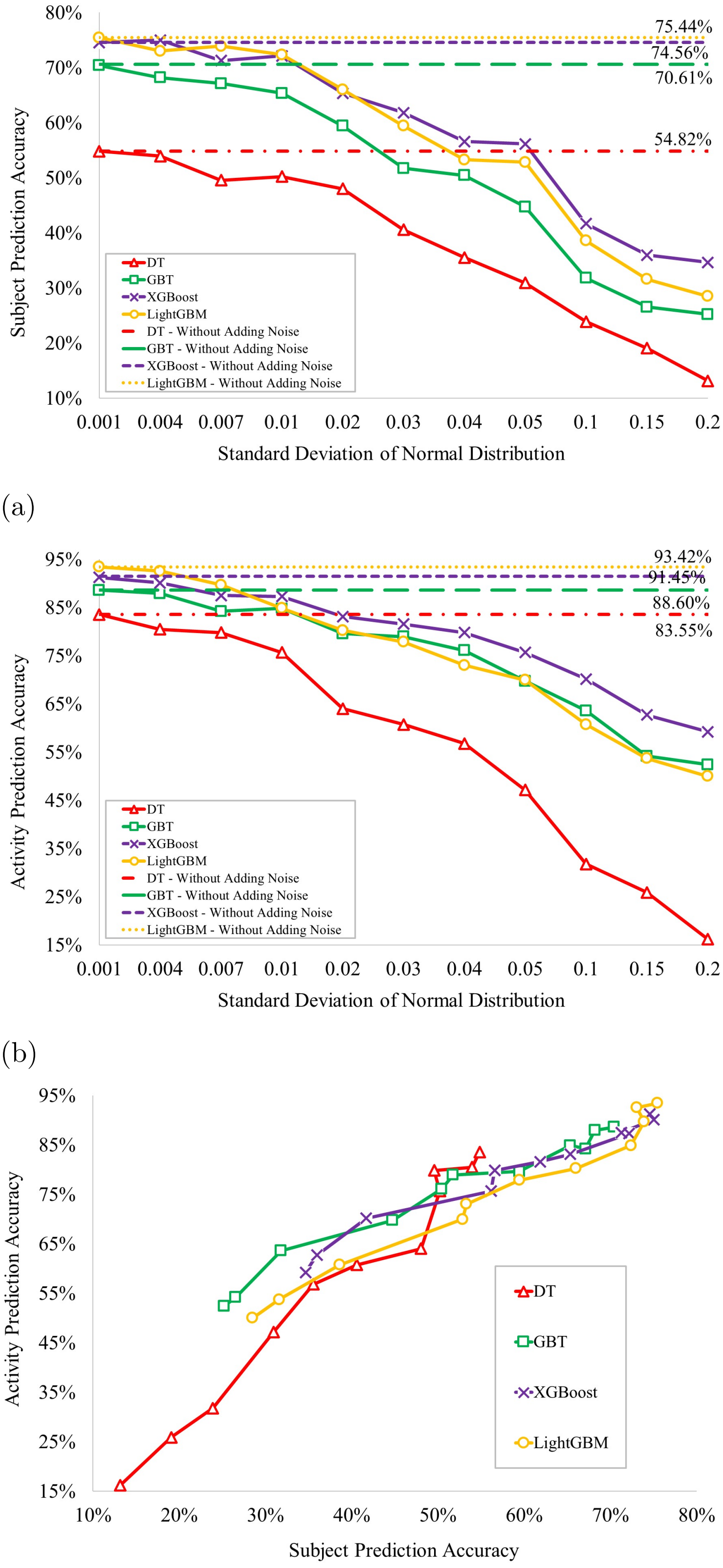


Figure 5.10 Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to all features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models.

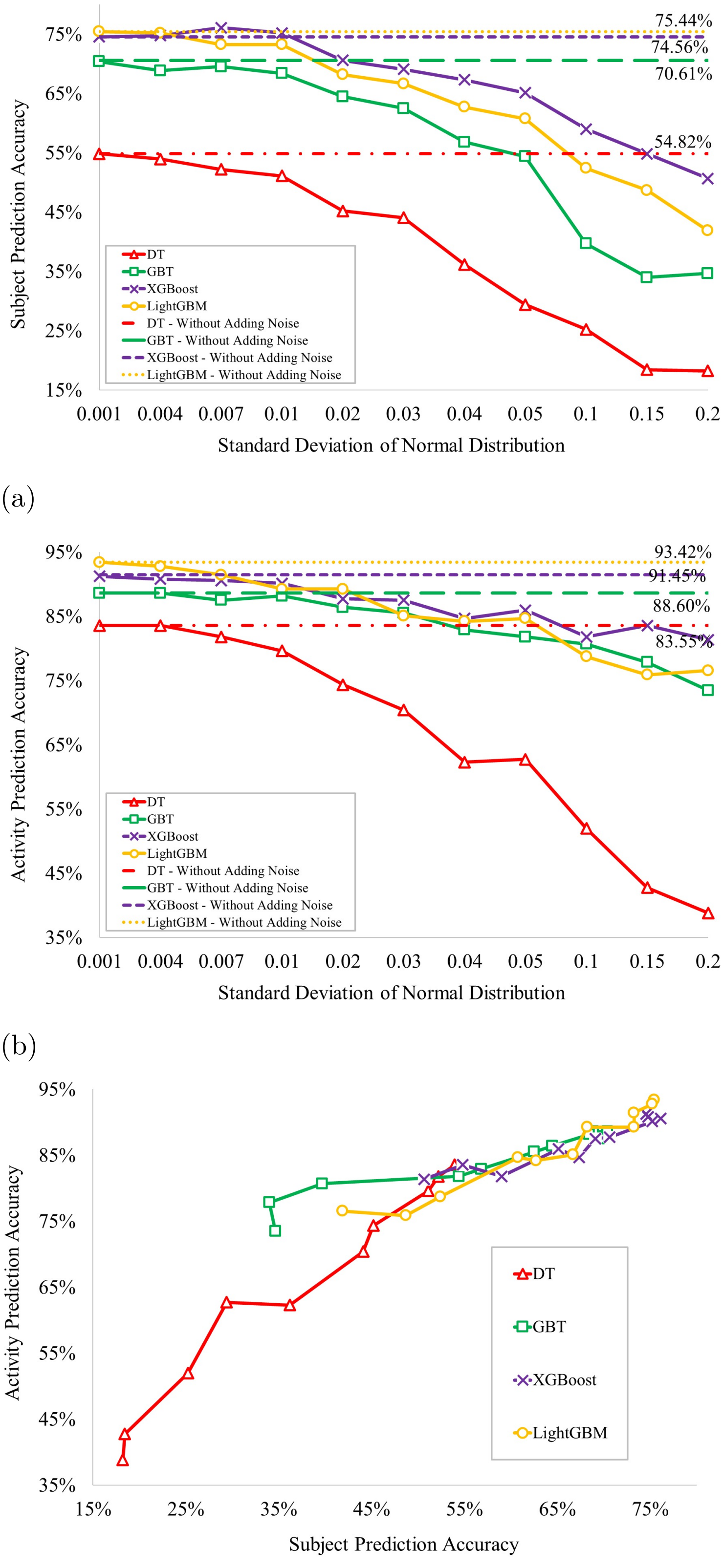


(a)

(b)

(c)

Figure 5.11 Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to random 75% of features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models.

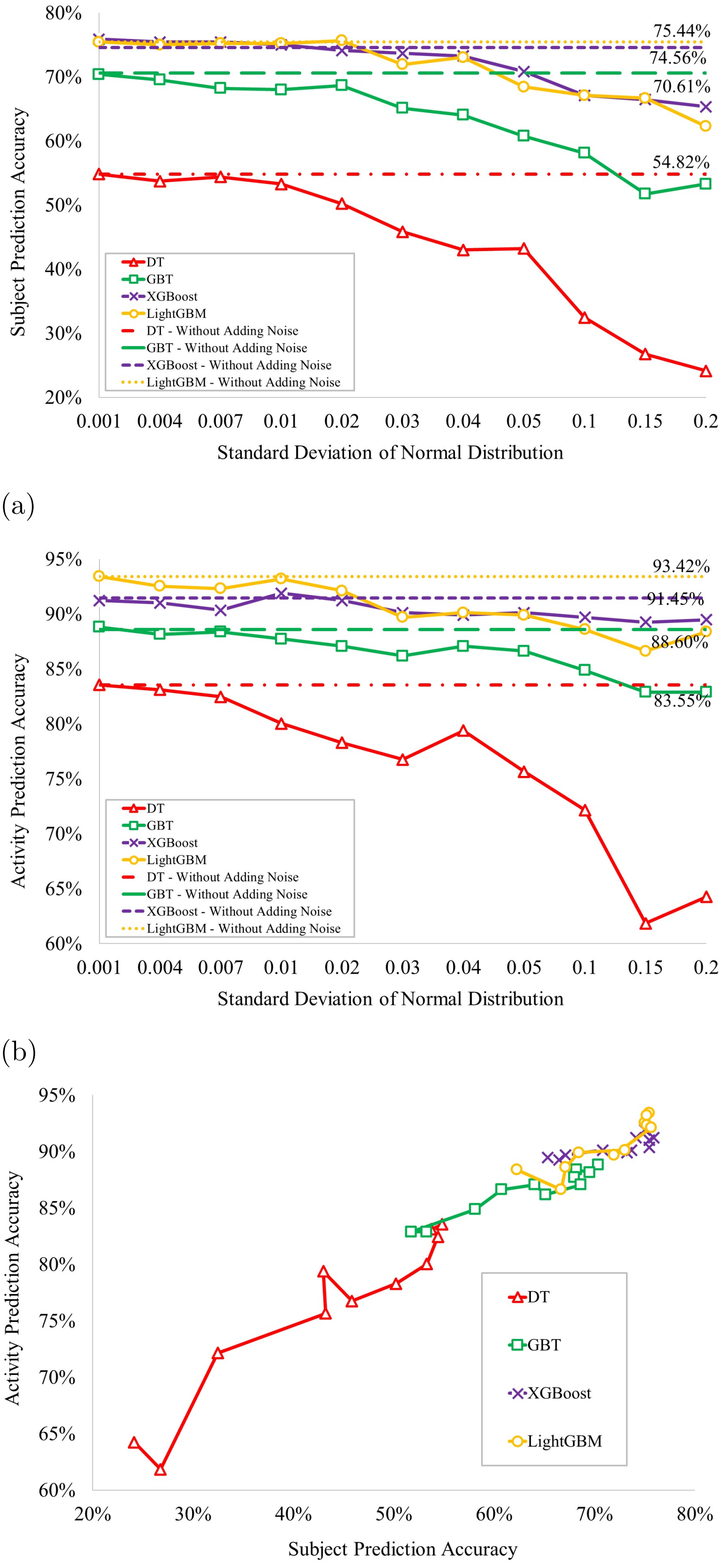


(a)

(b)

(c)

Figure 5.12 Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to random 50% of features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models.



(a)

(b)

(c)

Figure 5.13 Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to random 25% of features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models.

features that capture movement patterns across multiple sensors. Reducing the feature set diminishes the overall information available to activity models, leading to a moderate accuracy drop. However, this reduction also limits the extent to which noise is applied to activity-relevant features, mitigating the negative impact of noise. Moreover, the likelihood of retaining the 17 shared features decreases as the subset is reduced, lessening interference between subject and activity classification tasks and partially preserving activity prediction accuracy.

These observations indicate a trade-off inherent in feature subset selection: while subject prediction accuracy benefits from a higher relative representation of subject-relevant features and reduced noise interference, activity prediction accuracy experiences a smaller decline due to the decreased cross-task interference and noise exposure.

The findings in RFNI underscore the importance of designing noise injection strategies that selectively target features based on their relevance to specific tasks. This approach minimizes collateral damage to the primary objective—activity classification—while effectively mitigating subject inference attacks. Building on this idea, the next scenario explores a targeted strategy: the Subject-Specific Feature Noise Injection Defense Mechanism (SFNI).

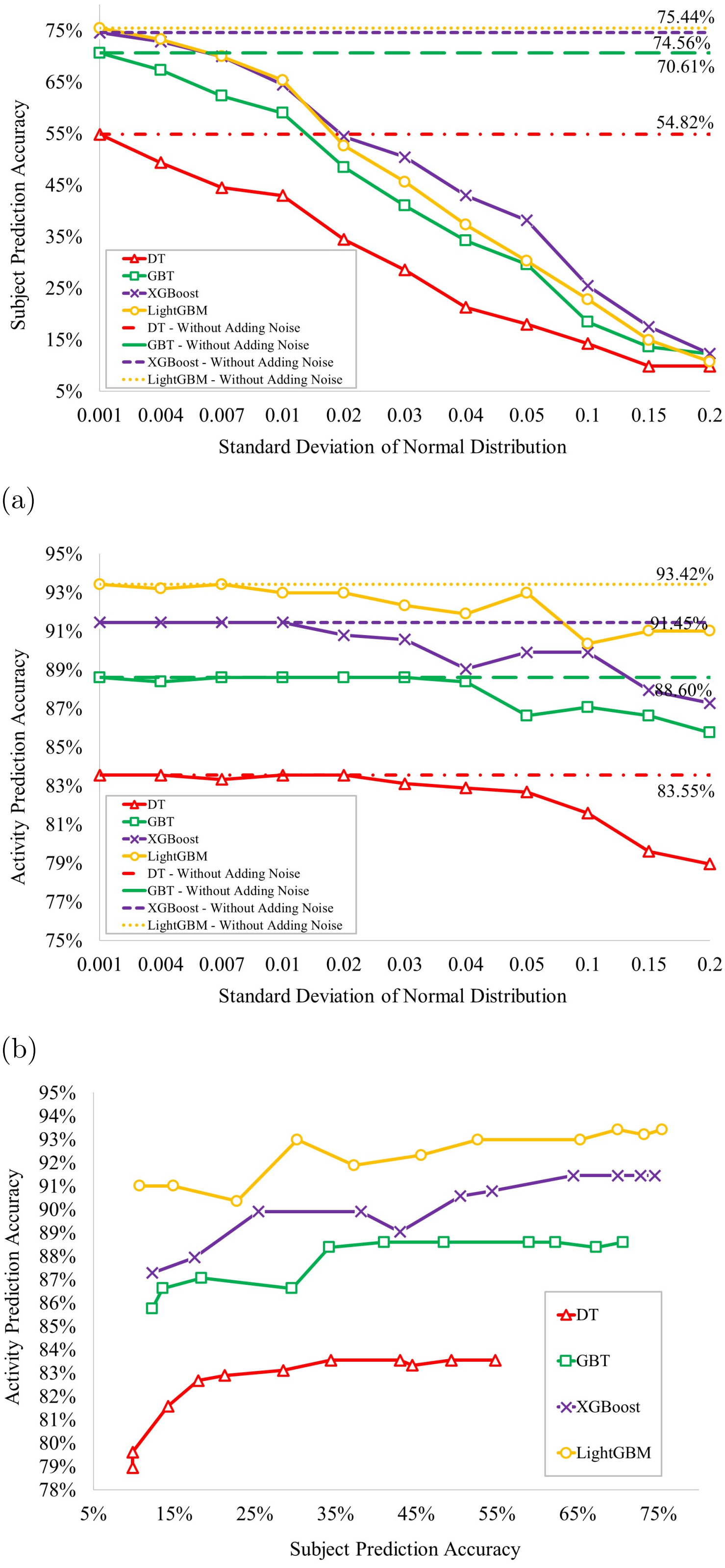
5.2.1.2 Subject-Specific Feature Noise Injection Defense Mechanism (SFNI)

In this scenario, the defense mechanism possesses additional knowledge about the features compared to the RFNI strategy. Instead of having access to all 295 features, the defender identifies the 156 features most critical for subject classification. Noise is applied exclusively to these features to maximize the reduction in subject prediction accuracy.

The changes in accuracy for the four activity and subject prediction models as a function of the applied noise are depicted in Figures 5.14a and 5.14b. As these figures demonstrate, subject prediction accuracy decreases significantly with increasing noise, achieving the desired outcome. Meanwhile, the impact on activity prediction accuracy is less pronounced compared to the previous scenario, effectively meeting both objectives. Figure 5.14c further highlights the trade-off between activity and subject prediction accuracies, showing that activity prediction accuracy is better preserved as subject prediction accuracy declines.

The results reveal three key observations:

Impact on Subject Prediction Accuracy: By focusing noise injection on features that are explicitly significant for subject classification, this defense mechanism effectively



(a)

(b)

(c)

Figure 5.14 Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to subject-specific features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models.

disrupts the information most critical to subject prediction models. Unlike RFNI, which distributes noise randomly across all features, SFNI ensures that the noise targets the core predictive elements for subject inference. This targeted approach leads to a more significant drop in subject prediction accuracy as noise intensity increases, making it more challenging for the attacker to infer subject identities.

Impact on Activity Prediction Accuracy: The effect of noise on activity prediction models is mitigated due to the selective application of noise. Since noise is applied only to the 156 subject-relevant features, the remaining features, including those critical for activity classification, remain largely unaffected. Additionally, the reduced overlap between subject- and activity-relevant features further minimizes the unintended impact of noise on activity prediction. This targeted strategy preserves the integrity of activity prediction models more effectively than RFNI, which applies noise indiscriminately across all features.

Trade-Off Between Objectives: Figure 5.14c illustrates that the SFNI mechanism achieves a more favorable trade-off between subject and activity prediction accuracies compared to RFNI. As subject prediction accuracy decreases due to noise, activity prediction accuracy experiences a much smaller decline. This result underscores the effectiveness of targeted noise application in reducing subject inference attacks while maintaining the primary functionality of activity classification systems.

While SFNI demonstrates a significant improvement in balancing the objectives of subject prediction reduction and activity prediction preservation, it still applies noise to features that are partially shared between the two tasks. This overlap can lead to unnecessary degradation in activity prediction accuracy. To address this, the next scenario introduces the Subject-Only Feature Noise Injection Defense Mechanism (SOFNI), which eliminates this overlap by isolating and targeting only the 139 features specific to subject classification. This refinement further minimizes collateral effects on activity prediction while maintaining robust protection against subject inference attacks.

5.2.1.3 Subject-Only Feature Noise Injection Defense Mechanism (SOFNI)

In the third scenario, the defense system leverages even more detailed knowledge about the features. This defender identifies both the 156 significant features for subject classification and the 156 significant features for activity classification. By categorizing features based on their relevance to each classification task, the defense mechanism isolates the 139 features that are exclusively important for subject classification, excluding the 17 features shared by both tasks. Noise is applied solely to these 139 features, ensuring that the features critical for activity classification remain entirely unaffected.

The changes in accuracy for the four prediction models as a function of the applied noise are shown in Figures 5.15a and 5.15b.

The results highlight the following key observations:

Impact on Subject Prediction Accuracy: Applying noise exclusively to the 139 subject-specific features ensures that the disruption directly targets the critical predictive elements for subject inference. As noise intensity increases, subject prediction accuracy declines significantly across all models, eventually reaching levels desired by the defense mechanism. This result underscores the effectiveness of precise noise targeting, as it maximizes the disruption to subject prediction models without diluting the noise effect by spreading it across irrelevant or redundant features.

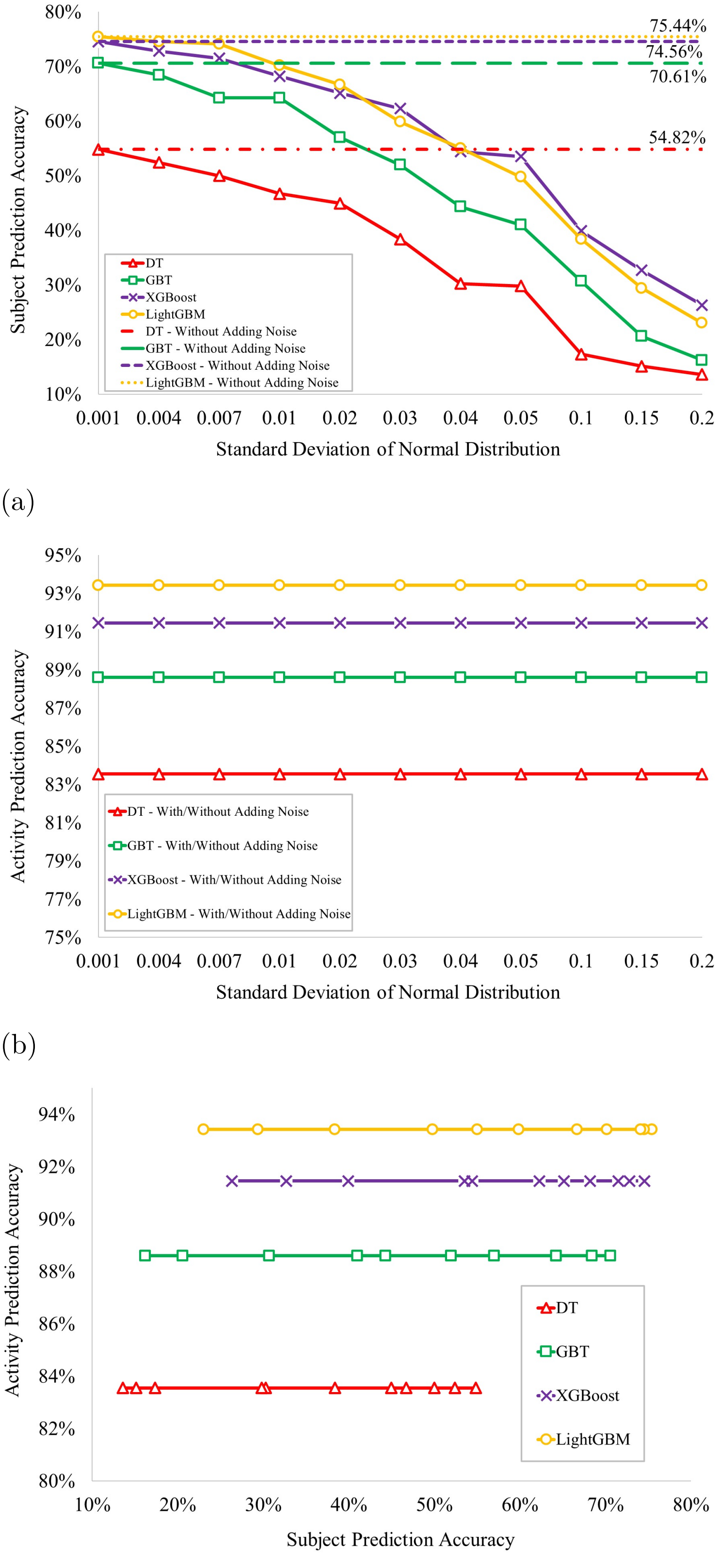
Impact on Activity Prediction Accuracy: Since no noise is applied to the 156 features significant for activity classification, activity prediction accuracy remains unaffected. Unlike RFNI and SFNI, where noise inadvertently impacts activity-relevant features due to overlap, the SOFNI approach eliminates any collateral effects. This precise targeting ensures that the primary objective of activity classification remains intact, even under increasing noise levels.

Trade-Off Between Objectives: Figure 5.15c illustrates the stability of activity prediction accuracy in contrast to the significant drop in subject prediction accuracy as noise intensity increases. This result demonstrates the superiority of SOFNI in balancing the dual objectives of preserving activity prediction accuracy and reducing subject prediction accuracy. Unlike previous scenarios, SOFNI achieves an ideal balance, where activity prediction models are entirely unaffected while subject prediction models are effectively disrupted.

Why Do the Results Occur This Way?

The outcomes observed in Figures 5.15a, 5.15b, and 5.15c can be attributed to the following factors:

Precise Targeting of Subject-Specific Features: By isolating the 139 features exclusive to subject classification, SOFNI ensures that noise is applied only to the features directly contributing to subject prediction. This precision eliminates the dilution of noise observed in RFNI and SFNI, where noise was either randomly distributed or partially affected overlapping features. Consequently, subject prediction models experience a more



(a)

(b)

(c)

Figure 5.15 Accuracy of (a) Subject prediction models and (b) Activity prediction models with and without noise of specific intensities added to subject-only features, along with (c) a trade-off analysis diagram showing the performance relationship between Subject and Activity prediction models.

substantial accuracy drop with increasing noise.

No Overlap Between Affected Features: The absence of overlap between the noisy features (subject-specific) and the unaffected features (activity-specific) ensures that activity prediction models remain entirely shielded from the defense mechanism's impact. This separation of feature sets prevents any unintended degradation in activity prediction accuracy, a significant improvement over the previous scenarios.

Enhanced Trade-Off Management: The trade-off between subject and activity prediction accuracies is more effectively managed in SOFNI due to its exclusive focus on subject-specific features. By leaving activity-critical features intact, the defense mechanism achieves a near-perfect balance, significantly reducing subject inference accuracy without compromising the primary functionality of activity classification systems.

SOFNI represents the most refined approach among the three scenarios, achieving the dual objectives of robust defense against subject inference attacks and uncompromised activity prediction accuracy. This approach demonstrates the importance of leveraging detailed feature knowledge in designing targeted defense mechanisms.

5.2.1.4 Comparison of the Three Features-Based Defense Mechanisms

From the first to the third scenario, the results progressively align more closely with the intended goals. In the third scenario, the ideal outcome is achieved: a substantial reduction in subject prediction accuracy with no change in activity prediction accuracy. As previously noted, the LightGBM model consistently outperforms the other three algorithms. For this model, Figure 5.16 illustrates the changes in subject prediction accuracy relative to activity prediction accuracy as noise intensity increases.

As seen in Figure 5.16, the subject prediction accuracy decreases by up to 10% in the first and second defense systems, which is better than the third one, where the reduction is around 20%. However, in the first defender, applying noise to all features significantly reduces activity prediction accuracy. In contrast, the second defender shows a much better performance, with only about a 2% reduction in activity prediction accuracy. The third defender, while causing no reduction in activity prediction accuracy, does not reduce subject prediction accuracy as effectively as the second defense mechanism (by about 10% less).

Thus, the second defender offers the best trade-off between the two objectives—maximizing subject prediction accuracy reduction while minimally impacting activity prediction accuracy—demonstrating better overall performance compared to the

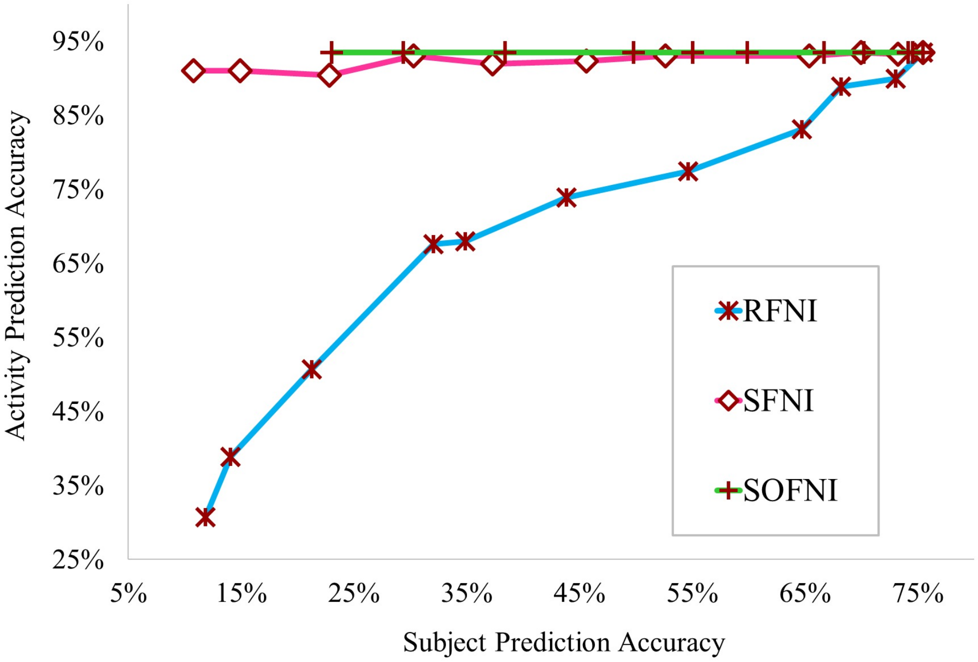


Figure 5.16 Impact of noise intensity on subject prediction accuracy and activity prediction accuracy across the three defense mechanisms for LightGBM models.

other two.

5.2.2 Raw-Sensor-Data-Based Defense Mechanisms

Although the feature-based defenders exhibit excellent performance, they are less realistic in practical applications. Depending on the microcontroller hardware used for sensor installation, raw sensor readouts are typically more accessible, and real-time computation and supplying large number of features is often infeasible. Thus, feeding defense mechanisms with features instead of raw sensor data is not practical. For this reason, this section focuses on designing defense systems that directly manipulate raw sensor data.

5.2.2.1 Uniform Noise-Based Defense Mechanisms

Uniform noise-based defense mechanisms provide a straightforward and effective approach to reduce the accuracy of Subject prediction models. By introducing noise uniformly across all sensor axes, these methods aim to disrupt predictive models' ability to identify sensitive information. This section introduces two variations of uniform noise-based defenses: Uni-SANI, which applies random uniform noise intensities, and OptUni-SANI, which fine-tunes noise intensity to minimize Subject prediction accuracy while preserving Activity prediction accuracy as much as possible.

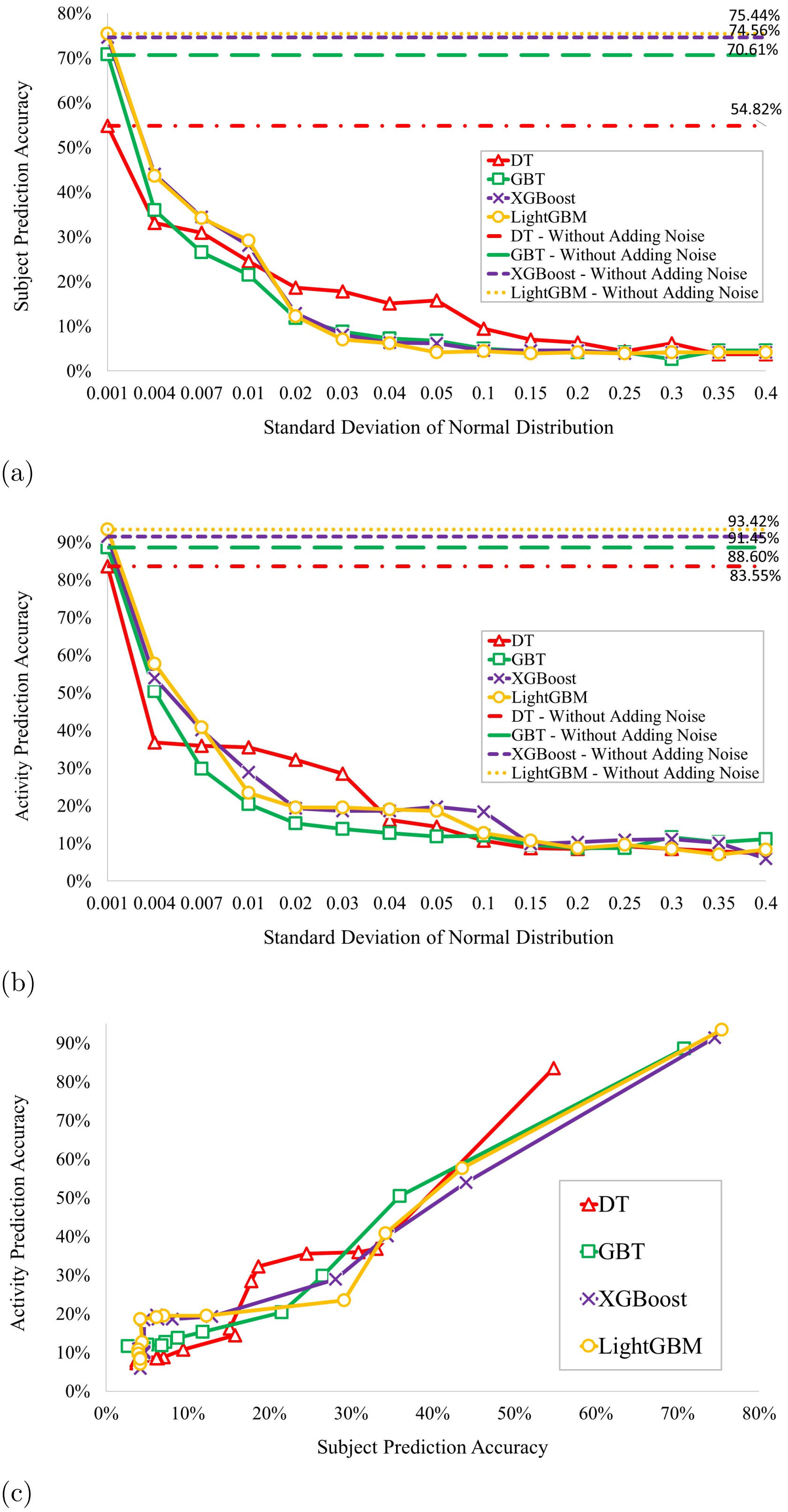
5.2.2.1.1 Uniform Sensor-Axis Data Noise Injection Defense Mechanism (Uni-SANI) The simplest and most cost-effective method to introduce noise into raw sensor data is to apply noise of uniform intensity across all 9 sensor-axes. For designing this defense system, it suffices for the defender to access readings from all three axes of the three available sensors. The defender generates random noise with varying intensities, adds it to the readings, and transmits the noisy data to the cloud service. Figures 5.17a and 5.17b illustrate how predictive models respond to this defense system's behavior under different noise intensities.

As observed, both the accuracy of the four Subject prediction models and the four Activity prediction models decline more sharply as noise intensity increases. Even with a slight increase in noise intensity, a significant drop in the models' accuracy is evident.

Figure 5.17c clearly depict how the accuracy of the Subject prediction models changes in comparison to the Activity prediction models. While the accuracy of the Subject models decreases well below our desired threshold (e.g., below 10%), the accuracy of the Activity models also declines significantly. This steep drop in Activity prediction accuracy is undesirable, as our objective is to maintain Activity prediction accuracy or allow only minimal reductions.

5.2.2.1.2 Optimized Uniform Sensor-Axis Data Noise Injection Defense Mechanism (OptUni-SANI) To mitigate the steep decline in the accuracy of Activity prediction models, we replace random noise intensity with a search for the minimum noise intensity that optimizes results when applied to all 9 sensor-axes. The optimization algorithm we use to determine the minimum standard deviation (σ) of normally distributed noise is the binary search algorithm. Among the four predictive models, we select only LightGBM, as it demonstrates higher accuracy compared to the other three models. By setting a threshold value ($\alpha = 40$) as the maximum acceptable accuracy for Subject prediction models, we proceed with the binary search for the minimum σ within the range [0, 0.5], starting with an initial value of 0.25. The optimization problem is defined as follows:

$$\begin{aligned} &\text{minimize} \quad \sigma \\ &\text{subject to} \quad \text{Acc}[\text{Subj. Pred.}, \text{Model}, \sigma] < \text{Threshold } \alpha \\ &\text{given} \quad \text{Model} = \{\text{LightGBM}\}, 0 < \sigma < 0.5, \alpha = 40 \end{aligned}$$



(a)

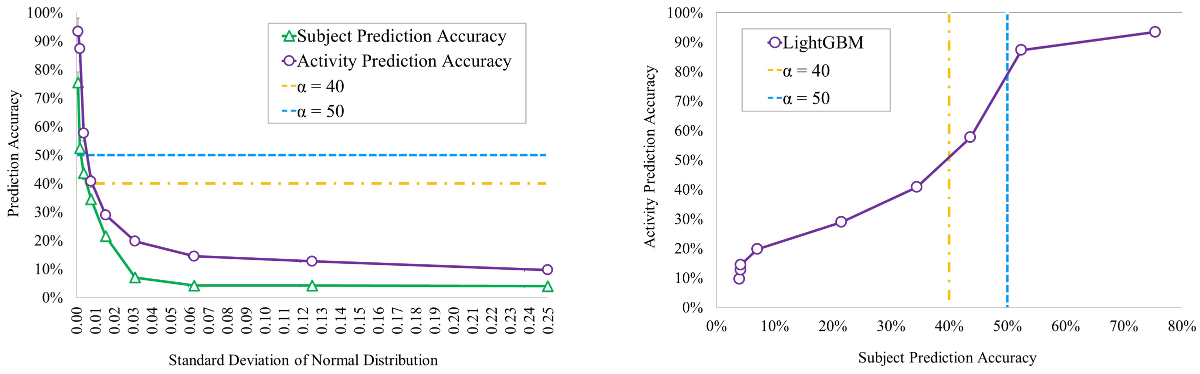
(b)

(c)

Figure 5.17 Effect of Uni-SANI on the accuracy decline of (a) Subject prediction models and (b) Activity prediction models with increasing noise intensity, while (c) shows the comparative changes in accuracy between Subject and Activity prediction models.

This optimized defense system requires access to predictive models to determine the effect of noise on the raw sensor data. The noisy data is tested on the Subject and Activity prediction models, and based on the accuracy results, the algorithm determines the next σ value for subsequent iterations.

Figure 5.18a illustrates how the accuracy of both Subject and Activity prediction models changes for the LightGBM model at different σ values. It can be observed that the search algorithm halves the σ value in each step. As σ decreases, the accuracy of both models increases, with the Subject prediction accuracy approaching the desired threshold. For instance, with a threshold $\alpha = 40$, the algorithm identifies the final noise standard deviation as $\sigma = 0.0078$. For $\alpha = 50$, the algorithm advances one more step to reach a final value of $\sigma = 0.0039$. Figure 5.18b further highlights how Subject prediction accuracy changes relative to Activity prediction accuracy, offering a clearer view for selecting an appropriate threshold value (α).



(a)

(b)

Figure 5.18 Performance of the OptUni-SANI defense mechanism. (a) Depicts changes in Subject and Activity prediction accuracy for the LightGBM model at varying noise standard deviation (σ) values during the binary search process. (b) Illustrates the relationship between Subject prediction accuracy and Activity prediction accuracy, aiding in the selection of the optimal threshold value (α).

5.2.2.2 Sensor-Axis-Specific Noise Analysis

In this analysis, noise is applied selectively to individual axes among the nine sensor-axes, rather than to all axes simultaneously. This approach aims to evaluate the unique significance of data from each sensor-axis in the predictive performance of both Activity and Subject classification models. By isolating each sensor-axis, it becomes possible to discern how the features extracted from these axes contribute to model accuracy and to understand the varying sensitivities of Subject and Activity prediction models to specific sensor data.

The number of significant features selected for classification based on either Activity or Subject often varies across sensor-axes. Even when the number of selected features is comparable, the features themselves may differ. Consequently, the influence of different sensor-axes on predictive accuracy is not uniform. For instance, data from certain sensor-axes may significantly affect Activity prediction models but have minimal or no impact on Subject prediction models, and vice versa. Furthermore, some axes may simultaneously influence both models, while others have negligible effects on either.

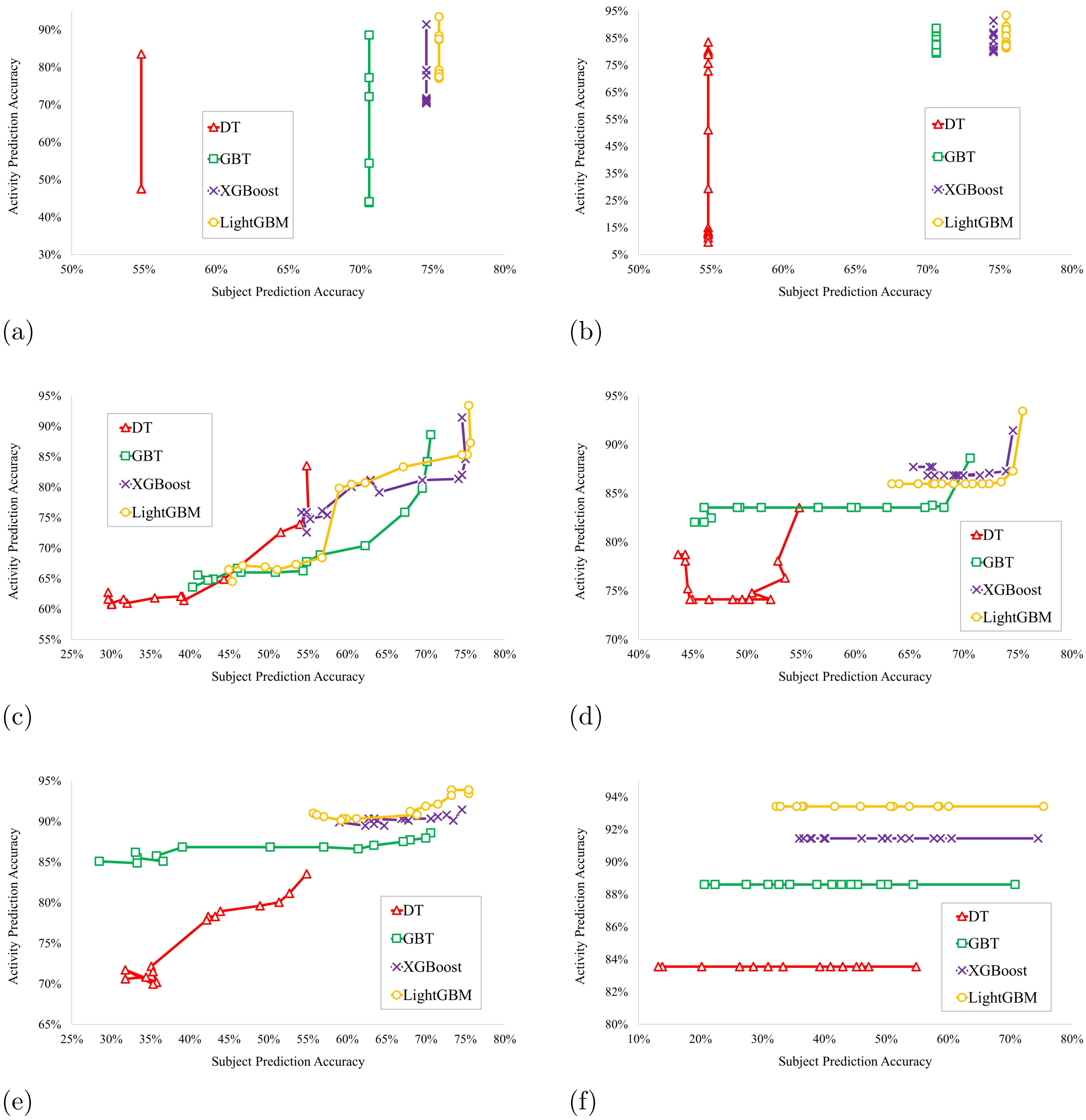
The results of this analysis highlight these distinctions. Figures 5.19a and 5.19b demonstrate that applying noise to the x- and y-axes of Acc2 has no effect on Subject prediction accuracy, indicating that the models did not select significant features from these axes. However, this same noise causes a notable decrease in Activity prediction accuracy, with reductions as high as 44% in the XGBoost model and 16% in the LightGBM model. Similarly, applying noise to the y-axis of Acc1 (Figure 5.19c) affects both Subject and Activity prediction models, showing sensitivity across both classification tasks.

The most intriguing results arise from applying noise to the x- and z-axes of Acc1 and the x-axis of the Gyro sensor (Figures 5.19d, 5.19e, and 5.19f). Here, noise significantly reduces Subject prediction accuracy while minimally affecting—or even preserving—the accuracy of Activity prediction models. For example, in the case of the Gyro sensor's x-axis, Subject prediction accuracy drops by approximately 43% to 50% across all models, while Activity prediction accuracy remains stable. These findings suggest that features selected from this sensor-axis are vital for Subject prediction but have minimal influence on Activity prediction, making it a prime candidate for further exploration.

The insights from Sensor-Axis-Specific Noise Analysis reveal how selectively applied noise can disrupt Subject prediction while preserving Activity prediction. This analysis not only identifies sensor-axes that hold asymmetric importance across classification tasks but also serves as a foundation for designing more targeted and effective noise-based defense mechanisms. Motivated by the findings of Sensor-Axis-Specific Noise Analysis, we introduce two novel mechanisms: MinNoise and AcConstrain, which build on the insights gained here to mitigate Subject inference attacks while safeguarding Activity prediction integrity.

5.2.2.3 Optimized Sensor-Axis-Specific Noise Injection Defense Mechanisms

The results from the previous defender led to the design of a third defense system. By adding noise to specific sensor-axis readings, we observed the behavior of Subject and Activity predictor models and how the accuracy of each predictor changes based on the



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.19 Results of Sensor-Axis-Specific Noise Analysis. (a) and (b) show how noise on the x- and y-axes of Acc2 impacts Activity but not Subject prediction accuracy. (c) Highlights accuracy reductions for both model types with noise on the y-axis of Acc1. (d), (e), and (f) depict the x- and z-axes of Acc1 and the x-axis of the Gyro sensor, where Subject prediction accuracy drops significantly while Activity prediction accuracy is preserved, with the Gyro x-axis showing the most desirable results.

sensor-axes affected by the noise. Consequently, we now have a better understanding of which sensor-axes should be targeted with noise to significantly reduce the accuracy of Subject predictors while ensuring minimal impact on Activity predictors.

In this new defending system, inspired by the results of the previous one, we select only two of the nine sensor-axes to apply noise: the z-axis of the Acc1 sensor and the x-axis of the Gyro sensor. Noise applied to these two sensor-axes demonstrated the highest reduction in the accuracy of Subject predictors while causing the least degradation in the accuracy of Activity predictors.

Applying noise with equal intensity across both sensor-axes is unlikely to yield optimal results. This was evident in the performance of the first defense mechanism (MinNoise), where noise with the same intensity was added to all nine sensor-axes, and optimization was needed to improve the results. Although the optimization brought relatively better results, there was still room for improvement. Applying different noise intensities to each of the nine sensor-axes and optimizing these intensities would provide even better results, albeit at a higher computational cost.

Thus, in this defense system, instead of directly applying noise with the same intensity, we apply two different noise intensities—each specific to one of the selected sensor-axes—and begin optimization simultaneously.

The selected optimization algorithm for this section is Bayesian search, which aims to find the optimal noise intensities σ_1 and σ_2 to apply to the z-axis readouts of the Acc1 sensor and the x-axis readouts of the Gyro sensor, respectively. The optimality criterion depends on the conditions we define. We solve the problem under two different conditions:

1. Minimizing the total noise intensities $\sigma_1 + \sigma_2$ required to reduce the Subject prediction model's accuracy below a predefined threshold (MinNoise).

2. Minimizing the accuracy of the Subject prediction model while ensuring the Activity prediction model's accuracy remains above a predefined threshold (AcConstrain).

For both scenarios, the LightGBM model is exclusively used for both Subject and Activity predictions. The details of each scenario are as follows.

5.2.2.3.1 Minimized Noise Optimization-Based Defense Mechanism (MinNoise) Under MinNoise, the objective is to minimize $\sigma_1 + \sigma_2$ such that the combined noise intensities reduce the Subject prediction model's accuracy below a specified threshold, α. The threshold α ranges from 15% to 60%, and for each value, Bayesian search is executed. Each execution involves 100 iterations, during which σ_1 and σ_2 are iteratively updated to minimize their sum while satisfying the condition that the Subject prediction accuracy falls below α.

The algorithm starts with initial values σ_1=0.25 and σ_2=0.25, and in subsequent iterations, it selects values for σ_1 and σ_2 from the range [0, 0.5]. This range provides sufficient flexibility for noise intensity selection. At the end of 100 iterations, the algorithm selects the σ_1 and σ_2 values that result in the minimum $\sigma_1 + \sigma_2$ while meeting the required condition. The optimization process for MinNoise can be expressed as follows:

$$\begin{aligned}
&\text{minimize} \quad \sigma_1 + \sigma_2 \\
&\text{subject to} \quad \text{Acc[Subj. Pred.,Model,}\sigma_1,\sigma_2] < \text{Threshold } \alpha \\
&\text{given} \quad \text{Model} = \{\text{LightGBM}\},\, 0 < \sigma_1, \sigma_2 < 0.5,\, \alpha = \{15\%, 20\%, 25\%, \ldots, 55\%, 60\%\}
\end{aligned}$$

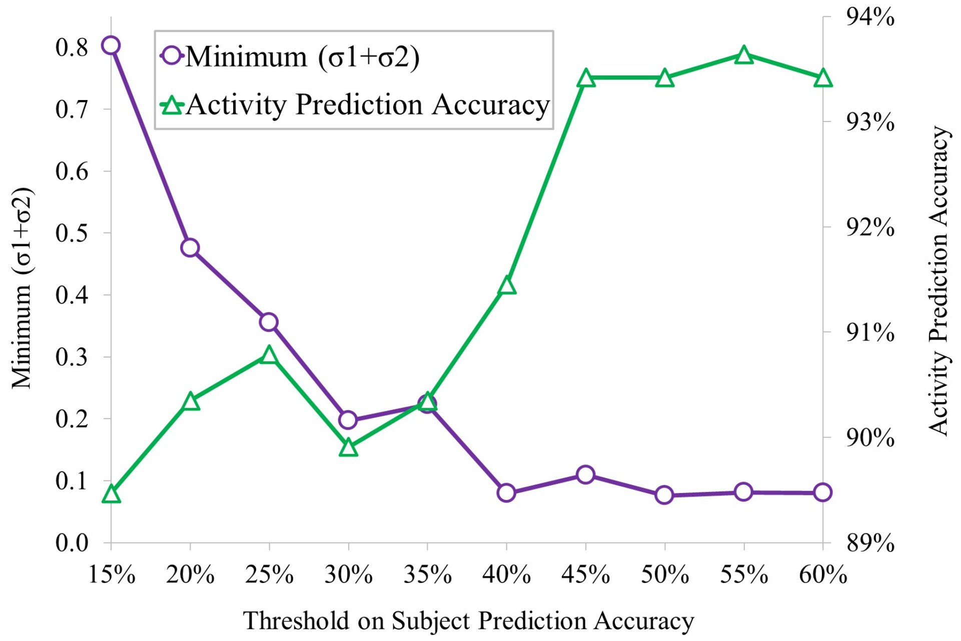
Figure 5.20a illustrates the minimum $\sigma_1 + \sigma_2$ values found by the algorithm for each threshold α. For instance, when α=15%, the algorithm identifies σ_1=0.496 and σ_2=0.3067, resulting in a total noise intensity of 0.8027. With these noise values, the Subject prediction model's accuracy is reduced to 14.47%, which is below the specified threshold. Notably, there is no constraint on the accuracy of the Activity prediction model in this scenario. Under the same conditions, the Activity prediction model maintains an accuracy of 89.47%.

While the algorithm also identifies other noise intensity combinations, such as σ_1=0.5 and σ_2=0.4895, which reduce the Subject prediction accuracy further (to 14.25%), the algorithm prioritizes minimizing the noise intensity sum and selects σ_1=0.496 and σ_2=0.3067 as the optimal solution for α=15%.

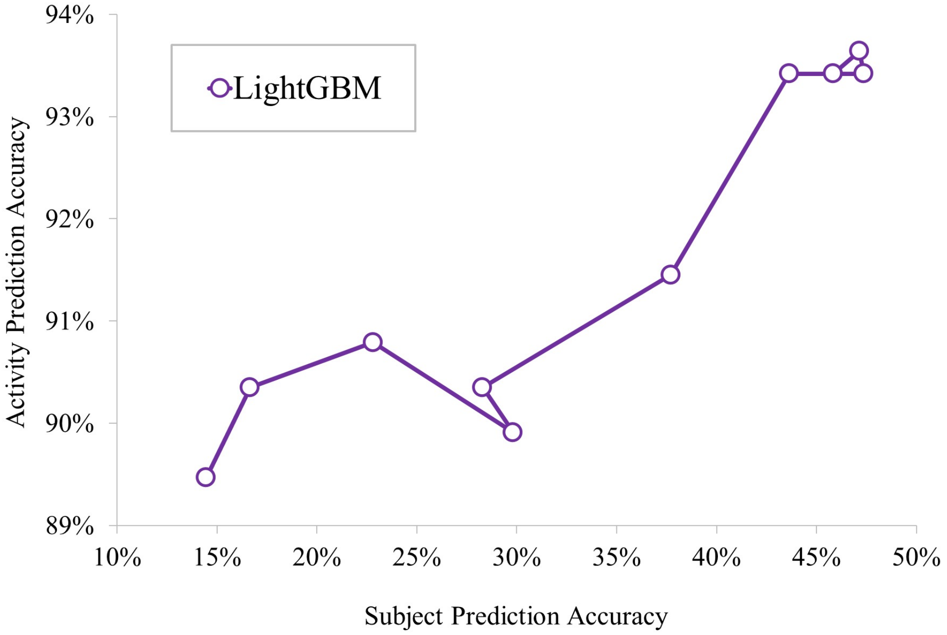
The results reveal that as the threshold α increases, the minimum noise intensities selected by the algorithm decrease. This is because lower noise intensities suffice to reduce the Subject prediction accuracy below higher thresholds. Moreover, as α increases, the Activity prediction accuracy experiences less degradation. For instance, the Activity prediction accuracy drops from 93.64% to 89.47% when α decreases, demonstrating effective performance under this optimization scenario.

Figure 5.20b illustrates the balance between the two models, revealing that a substantial drop in Subject prediction accuracy (from 47.37% to 14.47%) is accompanied by only a slight decline in Activity prediction accuracy (from 93.64% to 89.47%).

5.2.2.3.2 Accuracy-Constrained Optimization-Based Defense Mechanism (AcConstrain) Under AcConstrain, the objective is to minimize the Subject prediction model's accuracy while ensuring the Activity prediction model's accuracy remains above a threshold α. Unlike MinNoise, the priority here is to control the accuracy of



(a)



(b)

Figure 5.20 (a) Minimum total noise intensities ($\sigma_1 + \sigma_2$) required to reduce Subject prediction accuracy below various thresholds (α) under MinNoise. The results highlight the trade-off between noise intensity and Subject prediction accuracy, with lower thresholds requiring higher combined noise intensities. Activity prediction accuracy remains unconstrained in this scenario. (b) The balance between Subject and Activity prediction accuracies under MinNoise, showing that a significant reduction in Subject prediction accuracy leads to only a marginal decline in Activity prediction accuracy.

the Activity prediction model. The threshold α ranges from 84% to 94%, and Bayesian search is executed for each value.

Similar to MinNoise, each algorithm execution involves 100 iterations. During each iteration, the algorithm selects the σ_1 and σ_2 values from the range [0, 0.5] to minimize the Subject prediction accuracy while ensuring that the Activity prediction accuracy does not fall below α. The initial values are again set to σ_1=0.25 and σ_2=0.25.

The optimization process for AcConstrain can be expressed as follows:

$$\begin{aligned}
&\text{minimize} \quad \text{Subj. Pred. Acc.} \\
&\text{subject to} \quad \text{Acc}[\text{Act. Pred.}, \text{Model}, \sigma_1, \sigma_2] > \text{Threshold } \alpha \\
&\text{given} \quad \text{Model} = \{\text{LightGBM}\}, 0 < \sigma_1, \sigma_2 < 0.5, \alpha = \{84\%, 85\%, 86\%, \ldots, 93\%, 93.5\%, 94\%\}
\end{aligned}$$

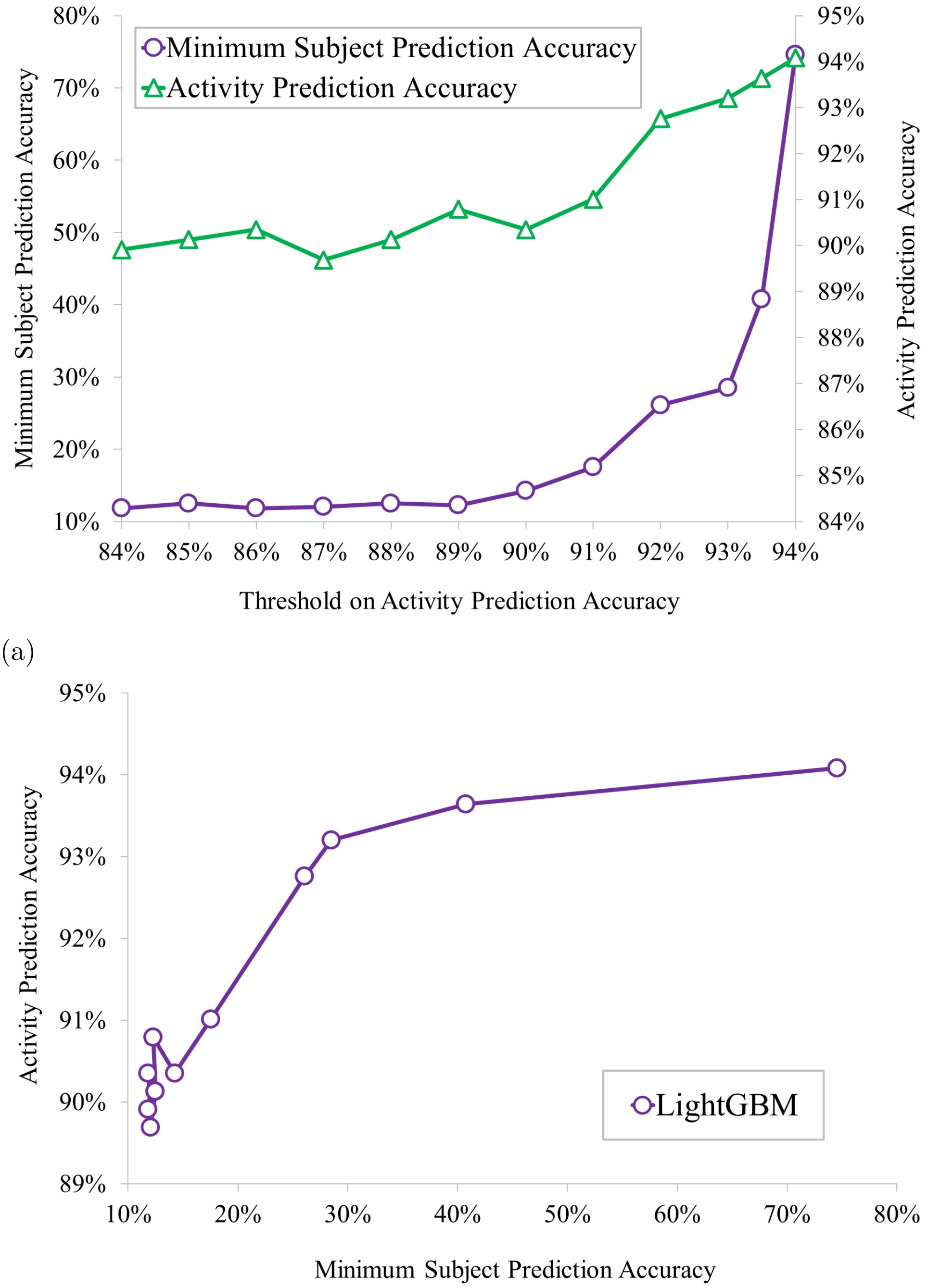
Figure 5.21a shows the minimum Subject prediction accuracies obtained by the algorithm for different α values. For instance, when α=93%, the algorithm finds σ_1=0.0109 and σ_2=0.4699, resulting in a Subject prediction accuracy of 28.51%, while maintaining the Activity prediction accuracy at 93.2%, above the threshold. Although other noise combinations, such as σ_1=0.0099 and σ_2=0.4702, yield lower total noise intensities, they are not selected as the final solution because they result in a slightly higher Subject prediction accuracy of 29.39%. The results indicate that as α increases, the minimum achievable Subject prediction accuracy also increases. This is because smaller noise intensities are selected to avoid significant degradation in Activity prediction accuracy. Consequently, the degradation in both Subject and Activity prediction accuracies is inversely proportional to α.

Figure 5.21b further highlights the trade-off between the two models, showing that even with a significant reduction in the Subject prediction accuracy (from 74.56% to 11.84%), the Activity prediction accuracy drops only marginally (from 94.08% to 89.91%).

5.2.2.4 Comparison of the Three Raw-Sensor-Data-Based Defense Mechanisms

The three proposed defense systems represent distinct approaches to introducing noise into raw sensor data, each with unique strengths and limitations.

Uni-SANI and its optimized version, OptUni-SANI, apply uniform noise across all sensor-



(a)

(b)

Figure 5.21 Performance of Optimized Sensor-Axis-Specific Noise Injection under Ac-Constrain defense mechanisms. (a) Minimum achievable Subject prediction accuracy for different Activity prediction thresholds (α), demonstrating the trade-off between the two models. (b) Comparative accuracy reductions for Subject and Activity prediction models, showing minimal impact on Activity prediction despite significant degradation in Subject prediction accuracy.

axes, achieving simplicity and broad effectiveness. While these approaches significantly reduce Subject prediction accuracy below desired thresholds, they also cause an undesirable drop in Activity prediction accuracy. The optimization of noise intensity using binary search in OptUni-SANI partially mitigates this issue but still struggles to preserve Activity prediction accuracy adequately.

MinNoise takes a more targeted approach by adding noise to individual sensor-axes. This design reveals the varying importance of different sensor-axes in predictive models. Results show that applying noise to specific axes, such as the x-axis of the Gyro sensor or the z-axis of the Acc1 sensor, achieves the desired reduction in Subject prediction accuracy while minimizing the impact on Activity prediction. However, identifying the most impactful axes requires additional analysis.

AcConstrain builds on the insights of MinNoise by optimizing noise intensities for two specific sensor-axes. Using Bayesian optimization, it balances the trade-off between minimizing Subject prediction accuracy and preserving Activity prediction accuracy. By tailoring noise intensities for each axis, this approach achieves the best performance among the three, effectively addressing both objectives with minimal computational overhead.

The performance comparison of all three defense mechanisms, under specific conditions, is depicted in Figure 5.22. The figure highlights the reductions in Subject prediction accuracy and the corresponding impact on Activity prediction accuracy for each mechanism, focusing on the LightGBM model. For Uni-SANI and OptUni-SANI, the comparison is based on uniform noise application. For MinNoise, the results are drawn from adding noise to the x-axis of the Gyro sensor, the sensor-axis with the most desirable result. Finally, for AcConstrain, both optimization conditions are considered, showcasing its ability to minimize Subject prediction accuracy while preserving Activity prediction accuracy.

In summary, Uni-SANI and its optimization offer simplicity but lack precision, while MinNoise and AcConstrain leverage axis-specific noise application to achieve more refined and effective outcomes. AcConstrain's optimization strategy makes it the most promising approach, providing significant reductions in Subject prediction accuracy with minimal impact on Activity prediction.

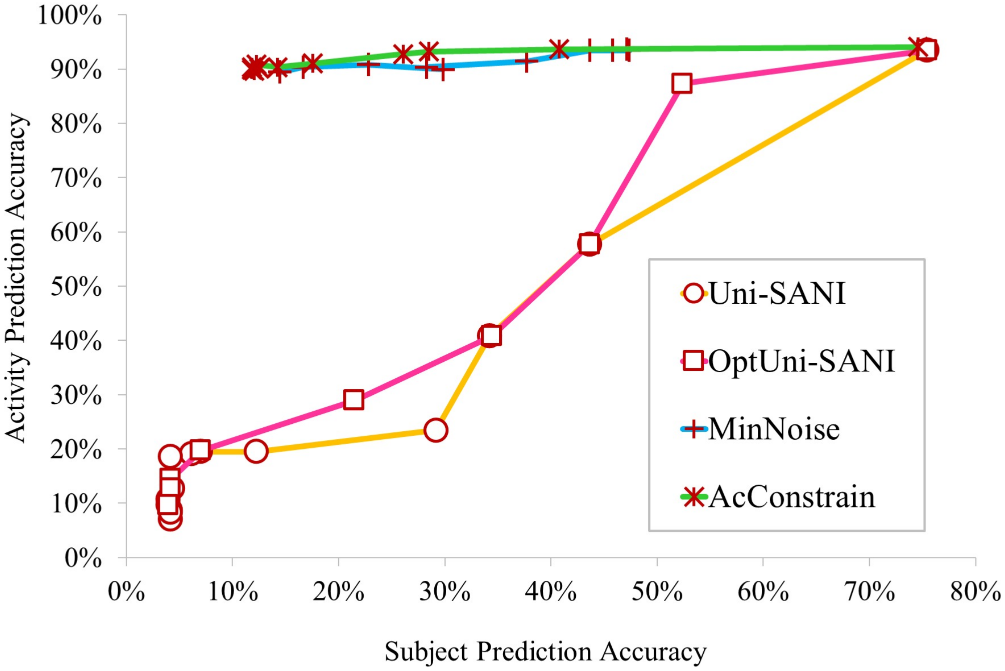


Figure 5.22 Comparison of the three raw-sensor-data-based defense mechanisms in terms of Subject and Activity prediction accuracy using the LightGBM model. The figure includes: (1) results for Uni-SANI and OptUni-SANI with uniform noise application, (2) MinNoise with noise applied to the x-axis of the Gyro sensor, and (3) AcConstrain for both optimization conditions. The comparison illustrates how each mechanism balances the trade-off between reducing Subject prediction accuracy and preserving Activity prediction accuracy.

6. CONCLUSION AND FUTURE WORK

This thesis has explored the hidden privacy risks embedded in two seemingly benign sensor data domains: ambient light emissions captured by smartphone sensors and biomechanical patterns in motor signal datasets. Through rigorous experimentation, we demonstrated how these data streams can be repurposed to infer sensitive user information, revealing critical gaps in current privacy protections. Our work underscores the dual-use nature of sensor technologies—while they enable valuable functionalities like adaptive screen brightness and fall detection, they simultaneously create unforeseen surveillance vectors that demand urgent attention.

The first part of this research introduced *LuxTrack*, a novel side-channel attack that exploits smartphone ambient light sensors to infer nearby laptop activities with over 80% accuracy—a 20% improvement over prior work. By developing a specialized Android app and collecting real-world data from human subjects, we established that subtle light variations from laptop screens carry sufficient information to distinguish between eight common activities, from PDF reading to video streaming. The effectiveness of this attack motivated the design of three practical countermeasures (binning, smoothing, and noise addition), which we empirically showed could reduce inference accuracy below 30% while preserving essential sensor functionality. These findings carry immediate implications for the W3C ALS standardization process and highlight the need for privacy-aware sensor architectures in mobile devices.

Parallel to this, our investigation of motor signal datasets uncovered equally severe privacy vulnerabilities. Using *SisFall* dataset as a case study, we demonstrated how ostensibly anonymized movement data can be reverse-engineered to re-identify individuals with 89.04% accuracy. This vulnerability stems from the inherent uniqueness of human biomechanical patterns, which persist across different activities. To mitigate this risk, we developed targeted defense mechanisms that strategically perturb either extracted features or raw sensor axes, achieving a 14.47% subject inference rate while maintaining over 95% utility for legitimate activity recognition tasks. These results challenge prevailing assumptions about dataset anonymization and call for fundamental changes in how sensor data is collected and shared, particularly in healthcare applications.

Looking ahead, several research directions emerge from this work. The attack surface of ambient light sensors could be further explored by combining ALS data with other smartphone sensors or expanding to IoT devices like smart bulbs, which increasingly incorporate similar light-sensing capabilities. For motor signals, developing real-time adaptive defenses that automatically adjust noise parameters based on contextual risk factors could bridge the gap between theoretical protections and practical deployment. Both domains would benefit from interdisciplinary solutions that integrate signal processing with cryptographic techniques, potentially enabling privacy-preserving sensor data sharing. At a broader level, our findings underscore the need for updated regulatory frameworks that address the unique challenges of sensor-based privacy invasions, ensuring that technological advancements in sensing capabilities are matched by corresponding safeguards for individual rights.

The methodologies and countermeasures presented in this thesis provide a foundation for rethinking sensor data privacy across multiple domains. By continuing to investigate these challenges while engaging with standardization bodies and policymakers, future research can help shape an ecosystem where sensor technologies fulfill their promise without compromising user trust.

BIBLIOGRAPHY

Abe, K., Sato, T., Watanabe, H., Hashizume, H., & Sugimoto, M. (2021). Smartphone positioning using an ambient light sensor and reflected visible light. In *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, (pp. 1–8). IEEE.

Al Nahian, M. J., Ghosh, T., Al Banna, M. H., Aseeri, M. A., Uddin, M. N., Ahmed, M. R., Mahmud, M., & Kaiser, M. S. (2021). Towards an accelerometer-based elderly fall detection system using cross-disciplinary time series features. *IEEE Access*, *9*, 39413–39431.

Al-qaness, M. A. A., Helmi, A. M., Dahou, A., & Elaziz, M. A. (2022). The applications of metaheuristics for human activity recognition and fall detection using wearable sensors: A comprehensive analysis. *Biosensors*, *12*(10), 821.

Bandt, C. & Pompe, B. (2002). Permutation entropy: a natural complexity measure for time series. *Physical Review Letters*, *88*(17), 174102.

Batista, G. E., Keogh, E. J., Tataw, O. M., & De Souza, V. M. (2014). Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, *28*, 634–669.

Benford, F. (1938). The law of anomalous numbers. *Proceedings of the American Philosophical Society*, 551–572.

Bet, P., Castro, P. C., & Ponti, M. A. (2019). Fall detection and fall risk assessment in older person using wearable sensors: A systematic review. *International Journal of Medical Informatics*, *130*, 103946.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control.* John Wiley & Sons.

Bruce Land. Complexity of a time series — hackaday project. `https://hackaday.io/project/707-complexity-of-a-time-series/details`. Accessed: 2025-06-15.

Casilari, E., Lora-Rivera, R., & García-Lagos, F. (2020). A study on the application of convolutional neural networks to fall detection evaluated with multiple public datasets. *Sensors*, *20*(5), 1466.

Casilari, E., Luque, R., & Morón, M.-J. (2015). Analysis of android device-based solutions for fall detection. *Sensors*, *15*(8), 17827–17894.

Chakraborty, S., Ouyang, W., & Srivastava, M. (2017). Lightspy: Optical eavesdropping on displays using light sensors on mobile devices. In *2017 IEEE International Conference on Big Data (Big Data)*, (pp. 2980–2989). IEEE.

Chatzaki, C., Pediaditis, M., Vavoulas, G., & Tsiknakis, M. (2017). Human daily activity and fall recognition using a smartphone's acceleration sensor. In C. Röcker, J. O'Donoghue, M. Ziefle, M. Helfert, & W. Molloy (Eds.), *Information and Communication Technologies for Ageing Well and e-Health* (pp. 100–118). Cham: Springer International Publishing.

Chelli, A. & Pätzold, M. (2019). A machine learning approach for fall detection and daily living activity recognition. *IEEE Access*, *7*, 38670–38687.

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, *307*, 72–77.

Coolen, M. (2019). Recognising gestures using time-series analysis. *Tilburg, the Netherlands.*

Delgado-Escaño, R., Castro, F. M., Cózar, J. R., Marín-Jiménez, M. J., Guil, N., & Casilari, E. (2020). A cross-dataset deep learning-based classifier for people fall detection and identification. *Computer Methods and Programs in Biomedicine*, *184*, 105265.

Delgado-Santos, P., Stragapede, G., Tolosana, R., Guest, R., Deravi, F., & Vera-Rodriguez, R. (2022). A survey of privacy vulnerabilities of mobile device sensors. *ACM Computing Surveys (CSUR)*, *54*(11s), 1–30.

Dhiman, C. & Vishwakarma, D. K. (2019). A review of state-of-the-art techniques for abnormal human activity recognition. *Engineering Applications of Artificial Intelligence*, *77*, 21–45.

Diamantaris, M., Marcantoni, F., Ioannidis, S., & Polakis, J. (2020). The seven deadly sins of the html5 webapi: A large-scale study on the risks of mobile sensor-based attacks. *ACM Transactions on Privacy and Security (TOPS)*, *23*(4), 1–31.

Frank, K., Vera Nadales, M., Robertson, P., & Pfeifer, T. (2010). Bayesian recognition of motion related activities with inertial sensors. In Bardram, J., Langheinrich, M., Truong, K., & Nixon, P. (Eds.), *Proceedings of the 12th ACM International Conference on Ubiquitous Computing Adjunct (UbiComp '10 Adjunct)*, (pp. 445–446)., Copenhagen, Denmark. ACM.

Galvão, Y. M., Ferreira, J., Albuquerque, V. A., Barros, P., & Fernandes, B. J. T. (2021). A multimodal approach using deep learning for fall detection. *Expert Systems with Applications*, *168*, 114226.

Henry, M. & Judge, G. (2019). Permutation entropy and information recovery in nonlinear dynamic economic time series. *Econometrics*, *7*(1), 10.

Hill, T. P. (1995). A statistical derivation of the significant-digit law. *Statistical Science*, 354–363.

Holmes, A., Desai, S., & Nahapetian, A. (2016). Luxleak: capturing computing activity using smart device ambient light sensors. In *Proceedings of the 2nd Workshop on Experiences in the Design and Implementation of Smart Objects*, (pp. 47–52).

Hussain, F., Hussain, F., Ehatisham-ul Haq, M., & Azam, M. A. (2019). Activity-aware fall detection and recognition based on wearable sensors. *IEEE Sensors Journal*, *19*(12), 4528–4536.

Hussain, M., Al-Haiqi, A., Zaidan, A. A., Zaidan, B. B., Kiah, M. M., Anuar, N. B., & Abdulnabi, M. (2016). The rise of keyloggers on smartphones: A survey and insight into motion-based tap inference attacks. *Pervasive and Mobile Computing*, *25*, 1–25.

Javed, A. R., Beg, M. O., Asim, M., Baker, T., & Al-Bayatti, A. H. (2020). Alphalogger: Detecting motion-based side-channel attack using smartphone keystrokes. *Journal of Ambient Intelligence and Humanized Computing*, 1–14.

Jokar, F., Azzopardi, G., & Palotti, J. (2023). Towards accurate and efficient sleep period detection using wearable devices. In *International Conference on Computer Analysis of Images and Patterns*, (pp. 43–54). Springer.

Katsidimas, I., Kotzakolios, T., Nikoletseas, S., Panagiotou, S. H., Timpilis, K., & Tsakonas, C. Impact Events for Structural Health Monitoring of a Plastic Thin Plate.

Katsidimas, I., Kotzakolios, T., Nikoletseas, S., Panagiotou, S. H., & Tsakonas, C. (2022). Smart objects: Impact localization powered by tinyml. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, (pp. 947–953).

Kwolek, B. & Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine*, *117*(3), 489–501.

Lempel, A. & Ziv, J. (1976). On the complexity of finite sequences. *IEEE Transactions on Information Theory*, *22*(1), 75–81.

Luna-Perejón, F., Domínguez-Morales, M. J., & Civit-Balcells, A. (2019). Wearable fall detector using recurrent neural networks. *Sensors*, *19*(22), 4885.

Maiti, A. & Jadliwala, M. (2019). Light ears: Information leakage via smart lights. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *3*(3), 1–27.

Marquardt, P., Verma, A., Carter, H., & Traynor, P. (2011). (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, (pp. 551–562).

Martínez-Villaseñor, L., Ponce, H., Brieva, J., Moya-Albor, E., Núñez-Martínez, J., &

Peñafort-Asturiano, C. (2019). Up-fall detection dataset: A multimodal approach. *Sensors*, *19*(9), 1988.

Mauldin, T. R., Canby, M. E., Metsis, V., Ngu, A. H. H., & Rivera, C. C. (2018). Smartfall: A smartwatch-based fall detection system using deep learning. *Sensors*, *18*(10), 3363.

Mazilu, S., Blanke, U., Calatroni, A., & Tröster, G. (2013). Low-power ambient sensing in smartphones for continuous semantic localization. In *4th International Joint Conference on Ambient Intelligence (AmI 2013), Dublin, Ireland*, (pp. 166–181). Springer.

Medrano, C., Igual, R., Plaza, I., & Castro, M. (2014). Detecting falls as novelties in acceleration patterns acquired with smartphones. *PloS ONE*, *9*(4), e94811.

Mehrnezhad, M., Makarouna, C., & Gray, D. (2022). Risks of mobile ambient sensors and user awareness, concerns, and preferences. In *Proceedings of the 2022 European Symposium on Usable Security*, (pp. 1–13).

Mitchell, M. (2009). *Complexity: A guided tour.* Oxford University Press.

Mrozek, D., Koczur, A., & Małysiak-Mrozek, B. (2020). Fall detection in older adults with mobile iot devices and machine learning in the cloud and on the edge. *Information Sciences*, *537*, 132–147.

Musci, M., De Martini, D., Blago, N., Facchinetti, T., & Piastra, M. (2020). Online fall detection using recurrent neural networks on smart wearable devices. *IEEE Transactions on Emerging Topics in Computing*, *9*(3), 1276–1289.

Narain, S., Vo-Huu, T. D., Block, K., & Noubir, G. (2016). Inferring user routes and locations using zero-permission mobile sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*, (pp. 397–413). IEEE.

Nguyen, K. A., Akram, R. N., Markantonakis, K., Luo, Z., & Watkins, C. (2019). Location tracking using smartphone accelerometer and magnetometer traces. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, (pp. 1–9).

Oppenheim, A. V., Schafer, R. W., & Buck, J. R. (1999). Discrete-time signal processing.

Pannurat, N., Thiemjarus, S., & Nantajeewarawat, E. (2014). Automatic fall monitoring: A review. *Sensors*, *14*(7), 12900–12936.

Richard, G. (2021). *Transfer Learning methods for temporal data.* PhD thesis, Université Paris-Saclay.

Richman, J. S. & Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, *278*(6), H2039–H2049.

Sabra, M., Maiti, A., & Jadliwala, M. (2018). Keystroke inference using ambient light sensor on wrist-wearables: a feasibility study. In *Proceedings of the 4th ACM Workshop on Wearable Systems and Applications*, (pp. 21–26).

Saleh, M., Abbas, M., & Le Jeannès, R. B. (2021). Fallalld: An open dataset of human falls and activities of daily living for classical and deep learning applications. *IEEE Sensors Journal*, *21*(2), 1849–1858.

Saleh, M. & Jeannès, R. L. B. (2019). Elderly fall detection using wearable sensors: A low cost highly accurate algorithm. *IEEE Sensors Journal*, *19*(8), 3156–3164.

Sarabia-Jácome, D., Usach, R., Palau, C. E., & Esteve, M. (2020). Highly-efficient fog-based deep learning aal fall detection system. *Internet of Things*, *11*, 100185.

Sato, T., Shimada, S., Murakami, H., Watanabe, H., Hashizume, H., & Sugimoto, M. (2021). Alisa: A visible-light positioning system using the ambient light sensor assembly in a smartphone. *IEEE Sensors Journal*, *22*(6), 4989–5000.

Schreiber, T. & Schmitz, A. (1997). Discrimination power of measures for nonlinearity in a time series. *Physical Review E*, *55*(5), 5443.

Schwittmann, L., Matkovic, V., Weis, T., et al. (2016). Video recognition using ambient light sensors. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, (pp. 1–9). IEEE.

Sikder, A. K., Petracca, G., Aksu, H., Jaeger, T., & Uluagac, A. S. (2021). A survey on sensor-based threats and attacks to smart devices and applications. *IEEE Communications Surveys & Tutorials*, *23*(2), 1125–1159.

Spreitzer, R. (2014). Pin skimming: exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, (pp. 51–62).

Sucerquia, A., López, J. D., & Vargas-Bonilla, J. F. (2017). Sisfall: A fall and movement dataset. *Sensors*, *17*(1), 198.

The Scipy community. scipy.signal.welch — scipy v0.14.0 reference guide. `https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.welch.html`. Accessed: 2025-06-15.

Usmani, S., Saboor, A., Haris, M., Khan, M. A., & Park, H. (2021). Latest research trends in fall detection and prevention using machine learning: A systematic review. *Sensors*, *21*(15), 5134.

Vavoulas, G., Chatzaki, C., Malliotakis, T., Pediaditis, M., & Tsiknakis, M. (2016). The mobiact dataset: Recognition of activities of daily living using smartphones. In *Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and e-Health (ICT4AWE)*, volume 2, (pp. 143–151). INSTICC, SciTePress.

Vavoulas, G., Pediaditis, M., Spanakis, E. G., & Tsiknakis, M. (2013). The mobifall dataset: An initial evaluation of fall detection algorithms using smartphones. In *2013 13th IEEE International Conference on Bioinformatics and Bioengineering (BIBE)*, (pp. 1–4). IEEE.

Vilarinho, T., Farshchian, B., Bajer, D. G., Dahl, O. H., Egge, I., Hegdal, S. S., & Weggersen, S. M. (2015). A combined smartphone and smartwatch fall detection system. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*, (pp. 1443–1448). IEEE.

Watt, S. J. & Politi, A. (2019). Permutation entropy revisited. *Chaos, Solitons & Fractals*, *120*, 95–99. Retrieved from https://www.sciencedirect.com/science/article/pii/S0960077919300335.

World Wide Web Consortium. W3c working draft - ambient light sensor.

Wu, X., Zheng, Y., Chu, C.-H., Cheng, L., & Kim, J. (2022). Applying deep learning technology for automatic fall detection using mobile sensors. *Biomedical Signal Processing and Control*, *72*, 103355.

Yacchirema, D., De Puga, J. S., Palau, C., & Esteve, M. (2018). Fall detection system for elderly people using iot and big data. *Procedia Computer Science*, *130*, 603–610. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

Yang, Z., Wu, C., Zhou, Z., Zhang, X., Wang, X., & Liu, Y. (2015). Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Computing Surveys (CSUR)*, *47*(3), 1–34.

Yentes, J. M., Hunt, N., Schmid, K. K., Kaipust, J. P., McGrath, D., & Stergiou, N. (2013). The appropriate use of approximate entropy and sample entropy with short data sets. *Annals of Biomedical Engineering*, *41*(2), 349–365.

Yu, X., Jang, J.-Y., & Xiong, S. (2021). A large-scale open motion dataset (kfall) and benchmark algorithms for detecting pre-impact fall of the elderly using wearable inertial sensors. *Frontiers in Aging Neuroscience*, *13*, 692865.

Yu, X., Qiu, H., & Xiong, S. (2020). A novel hybrid deep neural network to predict pre-impact fall for older people based on wearable inertial sensors. *Frontiers in Bioengineering and Biotechnology*, *8*, 63.