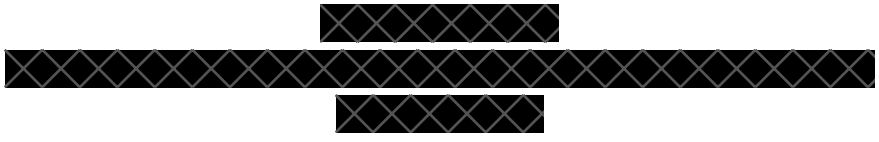
FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Dennis Goyal

ADVERSARIAL ATTACKS AND DEFENSE STRATEGIES IN MACHINE LEARNING BASED SENSOR FUSION SYSTEMS



ABSTRACT

This thesis research delves into identifying, analysing, and mitigating potential vulnerabilities in sensor fusion systems based on machine learning (ML) with a focus, on their applications in surveillance, security, intrusion detection and defence. These systems combine data from sensors like cameras, microphones, motion detectors, and thermal sensors, etc. The thesis first discusses the need for defences, formulating attack methodologies and demonstrating the ease of applying adversarial attacks to sensor fusion systems. The target systems are used for human action recognition tasks and collect data from Video, Infrared, depth cameras and inertial sensors. The thesis explores the vulnerability of these systems to attacks like Fast Gradient Sign Method (FGSM), Carlini&Wagner, and Temporal delay or shuffles. It also explores defence strategies and their effectiveness in safeguarding against these vulnerabilities. This analysis is not limited to the systems or datasets that are discussed in this study; it can be extended to other systems beyond the experimental scope of this study. The experiment analyses UTD-MHAD multi-modal dataset selected based on criteria such as sensor type, data complexity and real-world applicability. The objective is to facilitate the development of robust sensor fusion systems that are more secure, resilient, and dependable against these attacks.

Keywords: UTD-MHAD, FGSM, temporal misalignment, multi-modal data, defending attacks

TABLE OF CONTENTS

FOREWORD

This thesis was completed at the Oulu University Secure Programming Group (OUSPG) as part of a research project on robustness testing of artificial intelligence systems, funded by the Scientific Advisory Board for Defence (MATINE). I would like to express my sincere gratitude to my supervisor, Kimmo Halunen, for his invaluable guidance and support throughout this process. My heartfelt thanks also go to the entire research project group affiliated with this work, whose input and collaboration have been crucial in completing this thesis. Lastly, I would like to extend my deepest appreciation to my family for their unwavering support and encouragement during this journey.

Oulu, [redacted]
Dennis Goyal

LIST OF ABBREVIATIONS AND SYMBOLS

$\hat{\mathbf{x}}_k^-$	predicted state
$\hat{\mathbf{x}}_k$	updated state
$\mathbf{P}_k^-$, $\mathbf{P}_k$	predicted and updated error covariances
$\mathbf{u}_k$	control vector
$\mathbf{z}_k$	measurement vector
$\mathbf{K}_k$	Kalman Gain
$\mathbf{\Phi}$	Sensor Input transformation
$\mathbf{\Psi}$	Fusion Output
A	State transition matrix
B	Control input matrix
BIM	Basic Iterative Method
CNN	Convolutional Neural Network
EKF	Extended Kalman Filter
FGSM	Fast Gradient Sign Method
H	Measurement matrix
IMU	Inertial Measurement Unit
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori estimate
MSE	Mean Squared Error
PGD	Projected Gradient Descent
Q	Process noise covariance
R	Measurement noise covariance
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit Activation Function
S1, S2, S3	Sensor 1, Sensor 2, Sensor 3
SVM	Support Vector Machine
UTD-MHAD	University of Texas at Dallas Multimodal Human Action Dataset
ZOO	Zeroth-Order Optimisation
β	shape factor
ϵ_k	error signal at instant k
Φ_k	state transition matrix at instant k
θ	parameter vector
σ^2	variance
$\mathcal{L}$	Loss function
$\mathcal{N}$	Normal distribution
$\mathbf{h}$	Measurement function
$\mathbf{P}$	Estimation error covariance
$\mathbf{R}$	Sensor noise covariance
$\mathbf{W}$	Weight matrix
$\mathbf{y}$	Sensor output

1. INTRODUCTION

Sensor fusion systems are sophisticated frameworks that integrate data from multiple sources, such as cameras, radar, LiDAR (laser imaging, detection, and ranging), and other sensors, to create a comprehensive understanding of an environment or situation [1]. The principal motivation for multi-sensor data fusion is to improve the quality of the information output in a process known as synergy [2, 3]. The rationale behind sensor fusion techniques is to leverage the strengths of individual sensors while mitigating their respective weaknesses, thus enhancing the overall accuracy and reliability of the fused data [4].

Specifically, the system combines data from two or more sources in a way that generates a more consistent, accurate, and dependable understanding of the system [5]. Figure 1 outlines the key components of a fusion process in an autonomous system. The fusion process comes as the first component of an autonomous system to provide reliable signals for an action. It is designed to integrate information from multiple sensory sources across attributes, domain, and time. These systems are critical in various applications, such as autonomous vehicles, robotics, environmental monitoring, and healthcare [6, 7]. By effectively combining data from diverse sensors, these systems can improve decision-making processes, enhance system reliability, and increase the capability to handle uncertain environments [8, 9].

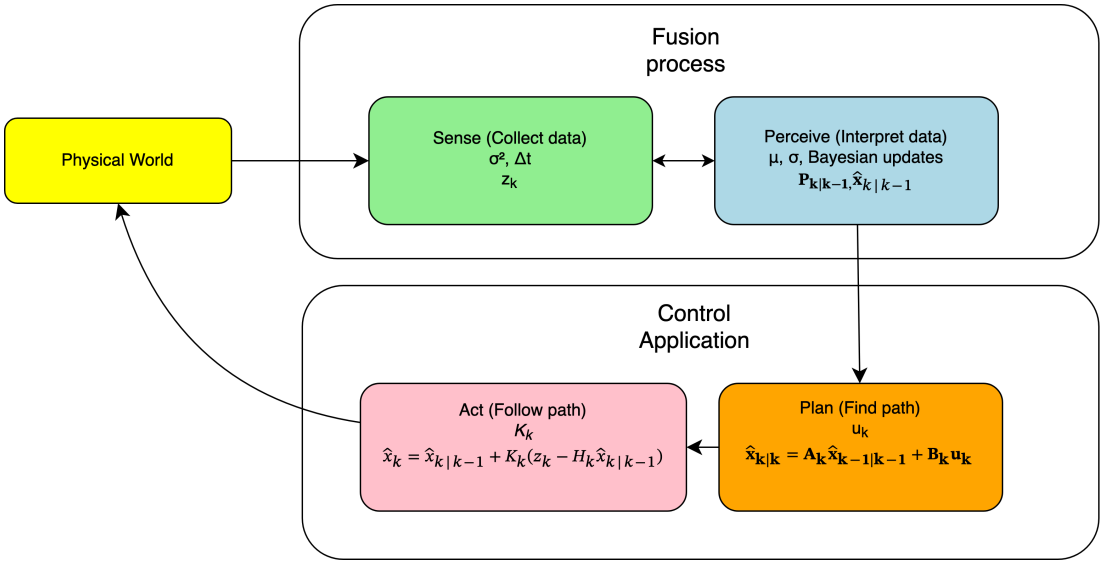


Figure 1. Process: Autonomous systems

The fusion process can be conducted at different levels, including raw data, feature, and decision levels, or hybrid (combining elements of data, feature, and decision-level fusion) depending on the complexity of the task and the desired outcome [3, 9].

In a typical multi-sensor fusion system, multiple sensors collect data about an object or an environment from different viewpoints, modalities, or spectra. Each sensor provides measurements that might be partial, noisy, or uncertain [1]. In the initial stages of sensor fusion, systems primarily used deterministic algorithms for integrating sensor data. Kalman filters, for instance, were extensively used in aerospace for navigation and tracking based on their linear dynamic models [10, 11]. They are known for its computational simplicity, low memory requirements, and robust noise reduction

capabilities, and are widely used for state estimation by extracting reliable information from noisy data. These filters effectively combined noisy sensor data from radar and sonar by predicting the state of a system over time and updating this prediction based on new measurements [12]

In addition to Kalman filters, early sensor fusion systems often relied on data association techniques, crucial in multi-target tracking scenarios, such as those encountered in radar surveillance. Extensions of the Kalman Filter for nonlinear systems involve linearisation or moment matching to approximate updates [13]. Extensions such as the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) have been developed to address some of these issues, but they still face challenges in highly dynamic and unpredictable environments [14].

Nevertheless, these methods are limited when dealing with systems characterised by highly non-Gaussian noise, as the assumption of Gaussian noise can significantly degrade performance. More advanced techniques for e.g. Particle Filters, for instance, use a set of samples (particles) to represent the probability distribution of the state, making them suitable for highly non-linear and non-Gaussian problems [15, 16]. Similarly, Bayesian Filters extend the principles of Kalman Filters by incorporating Bayesian inference, providing a more flexible framework for sensor fusion [17].

The advent of new and complex technologies, such as large data processing systems, has enabled the processing of complex (non-linear) data in greater volumes [18]. This limited the capability of traditional methods to produce more accurate results with additional information, leading to the introduction of Machine Learning (ML) in sensor fusion systems [19]. The infusion of ML into sensor fusion systems brought about an era of adaptive algorithms capable of dealing with non-linear, high-dimensional data in ways that traditional methods could not. ML-driven sensor fusion systems are at the forefront of technological advancement, particularly visible in the automation systems.

However, this complexity and reliance on ML introduce new challenges, particularly in preventing these systems from being misled by adversarial inputs designed to manipulate or confuse the algorithms [20]. While this integration enhances performance and expands application scope, it also brings new vulnerabilities. The increased exposure to adversarial attacks, through malicious inputs, can lead to system failures or misjudgements [21, 22].

They can alter sensor data to mislead the system, potentially causing catastrophic outcomes in critical applications like autonomous navigation and public safety. To defend, it may involve mitigation strategies such as adversarial training, where systems are exposed to a spectrum of attacks during the training phase, enhancing their resilience [20]. Additionally, implementing anomaly detection algorithms can aid in recognising and responding to unusual patterns indicative of an attack, reinforcing system defences [23, 22].

To mitigate the risks posed by adversarial attacks, research is continuously directed toward the development of resilient machine learning (ML) models. These efforts are crucial for the broader adoption of sensor fusion systems in sensitive areas. Effective defence against such attacks hinges on a deep understanding of the system's architecture and the dynamics between different sensor inputs, which is more complex compared to systems using single-modality data [21, 24]. To counter sophisticated attacks, it is essential to implement a layered security approach. This includes regular audits of the system, timely updates with the latest security patches, and engaging in

comprehensive testing scenarios that mimic potential attack vectors. These measures are vital for proactively strengthening system vulnerabilities and sealing off potential exploitation avenues [25]. Furthermore, by anticipating potential adversarial tactics, developers can proactively enhance the robustness of systems. Continuous research and adaptation of defensive strategies are imperative for maintaining the integrity and security of ML models against evolving threats [26].

This thesis focuses on the examining of the impact of adversarial attacks on ML sensor fusion systems, with the example of human-action recognition systems. This research is **vital** because it addresses the critical vulnerabilities and limitations of ML-based sensor fusion systems due to adversarial attacks, which can compromise the security and reliability of essential technologies in fields like autonomous systems and security applications. The research investigates how adversarial attacks, such as the Fast Gradient Sign Method (FGSM) [27] and Generative Adversarial Networks (AdvGAN) [28], compromise the integrity of these systems by exploiting vulnerabilities. By implementing some of these attacks on a variety of sensor modalities, including RGB cameras, depth sensors, and inertial sensors, the study highlights the susceptibility of ML fusion systems to perturbations that may appear imperceptible to the human eye but can significantly alter model predictions. Additionally, the thesis explores various defence strategies to enhance the robustness of these systems against such attacks. By integrating multiple defence mechanisms and evaluating their effectiveness, the research aims to contribute to the development of more secure and reliable sensor fusion systems, ensuring their resilience in critical applications such as surveillance, security, and autonomous systems.

1.1. Structure of the Thesis

The thesis is organised into several key sections to provide a comprehensive exploration of adversarial attacks and defence strategies in ML sensor fusion systems. The introductory chapter sets the stage by outlining the motivations, objectives, and scope of the research. Following this, the second chapter delves into the fundamental setup and evolution of multi-sensor fusion systems, it's drawbacks, limitations and highlighting the importance of sensor integration for enhanced accuracy and reliability in various applications. More importantly, it offers motivation by examining and discussing some limitations of current fusion systems and their defences, highlighting the significance of research in this field. The third chapter transitions into the core focus on adversarial attacks, looking at the background studies and related work, detailing different attack methodologies and their impacts on sensor fusion systems. This chapter also introduces defence mechanisms and strategies to mitigate these attacks, ensuring system robustness. The fourth chapter discusses the design and implementation of an experimental study on Human-Action Recognition systems, including the selection of datasets, the development of fusion models, attack techniques and their prevalence, and the defences to these attacks. The final sections of the thesis present the results, summarising key findings, discussing the implications of the research, and suggesting avenues for future work.

2. SENSOR FUSION SYSTEMS

Sensor fusion integrates data from multiple sensors to interpret the environment comprehensively, enhancing accuracy and reliability [29]. This is crucial for applications ranging from autonomous navigation to patient monitoring. In this topic, we will cover how sensor fusion systems have evolved, the introduction of ML into sensor fusion systems, and their applications in real-world scenarios.

2.1. Fundamental Setup of a Multi-Sensor Fusion System

A typical multi-sensor fusion system involves several key components and interactions. Say, Data input is taken from 3 sensors Sensor 1 (S1), Sensor 2 (S2), and Sensor 3 (S3). Each sensor collects data over time, denoted as $z_1(t), z_2(t), z_3(t)$, where t indicates the t^{th} measurement. Each sensor output is modelled mathematically and the fusion process can be done at any stage of pre-processing step or fusion stage see, 2.6.2. Figure 2 provides a detailed depiction of a typical multi-sensor fusion system.

For instance, Sensor 1 might measure temperature, leading to measurements denoted as $z_1(t)$, which include associated noise and inaccuracies.

- $z_i(t)$: Measurement from sensor i at time t.
- Φ_i: Represents the specific function or transformation applied to the raw data from sensor i, potentially encompassing calibration, scaling, or other preprocessing steps to prepare the data for fusion.
- We represent the fusion process by a function F that combines inputs $\Phi_1, \Phi_2, \Phi_3, \ldots$ into a comprehensive output Ψ.

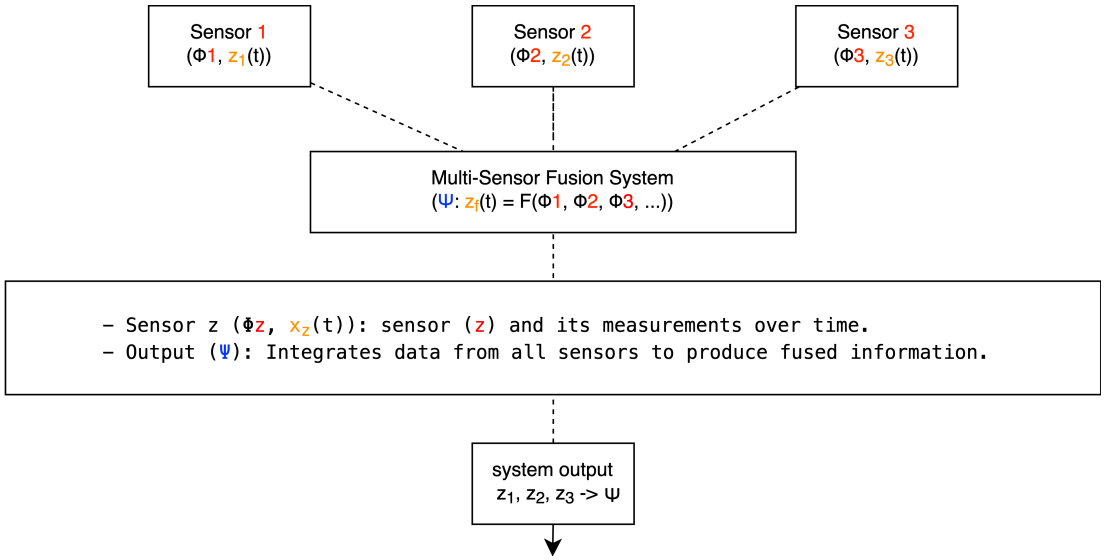


Figure 2. Fusion across attributes

The fused output at time t is denoted as Ψ, where $z_f(t)$ is a synthesised estimation based on all sensor inputs, aiming to reduce uncertainty and enhance reliability.

2.2. Advancements in Sensor Fusion and Machine Learning

Sensor fusion's evolution reflects advancements in computational power and algorithmic sophistication. Early fusion algorithms employed simple averaging or voting schemes [30]. Contemporary approaches utilise Bayesian methods, Kalman filters, and information theory to integrate heterogeneous (dissimilar) data sources, enabling real-time high-dimensional data processing [31]. For instance, combining visual data from cameras with radar and LiDAR data in autonomous vehicles allows for better object detection and situational awareness than relying on a single sensor type [17]. Extensions such as the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) have been developed to address some of these issues, but they still face challenges in highly dynamic and unpredictable environments [32].

Particle Filters, or Sequential Monte Carlo methods, represent a more adaptable approach suited for non-linear and non-Gaussian systems [15]. These algorithms work by generating a set of hypothetical states (particles) based on prior distributions, which are then updated as new data arrives, effectively capturing the posterior distribution of the system's state. This method is particularly valuable in scenarios where the system dynamics are complex or poorly defined, offering a robust alternative to traditional linear models. The flexibility and capability of Particle Filters to handle a diverse range of sensor fusion challenges mark their significance in the progression towards more sophisticated machine learning-based approaches in sensor integration.

The evolution of machine learning (ML) systems has also been significantly influenced by the development and subsequent mitigation of various adversarial attacks [33, 28]. These attacks have not only exposed vulnerabilities within ML frameworks but have also led to advancements in defensive strategies. One of the seminal moments in the history of ML security was the discovery of adversarial examples, where small, often imperceptible perturbations to input data could lead to erroneous outputs from ML models. Szegedy et al.'s research from 2013 made this discovery popular by proving that even cutting-edge neural networks were susceptible to these subtle changes [34]. It highlighted the vulnerable nature of ML models and the need for robust defence mechanisms, setting the stage for a wave of research focused on understanding and countering adversarial attacks. The challenges grew higher as ML systems began to be integrated into more complex applications, such as automation and sensor fusion systems. Automation systems were seen as particularly vulnerable due to their complexity and the diverse nature of their input data. Attacks on these systems often involved crafting adversarial inputs across different sources, making the detection and defence even more challenging [35].

The development of these advanced attacks drove a corresponding change in defensive techniques. One early approach was adversarial training, where models are trained not only on clean data but also on adversarial examples [35]. This method, proposed by Goodfellow et al., aimed to improve the robustness of ML models by exposing them to potential attacks during the training phase [27]. While effective to some extent, adversarial training alone is not enough, as it often required a significant increase in computational resources and could lead to overfitting to specific types of adversarial examples.

In response to the limitations of adversarial training, researchers have explored a variety of other defence mechanisms. One strategy is the development of detection

techniques designed to identify adversarial inputs before they could affect the model's performance. These techniques often relied on statistical analyses of input data to flag anomalies that might indicate an attack. For example, inputs to the model are checked against expected inputs to the model for which it was trained.

Additionally, techniques like defensive distillation, which Papernot et al. suggested, aim to lessen the sensitivity of models to small changes by training them to provide softer class probabilities [22]. This made it harder for adversarial examples to lead to wrong classifications. The use of ensemble methods, where multiple models are trained to make predictions and their outputs are combined to reach a consensus, helps in the diversity of different models to dilute the impact of any single adversarial example.

Generative Adversarial Networks (GANs) have emerged as a powerful tool in machine learning [36], particularly in the generation of artificial data that closely mimics real-world distributions. GANs consist of two neural networks—a generator and a discriminator—that are trained simultaneously in a competitive setting. The generator creates artificial data, while the discriminator evaluates its authenticity, leading to the generation of highly realistic data with training. This ability to produce artificial data has found applications across various domains, from image and video generation to data augmentation.

GANs, however, can also be employed to craft adversarial examples that are designed to deceive machine learning models [37]. A notable example is AdvGAN, which utilises the GAN architecture to generate adversarial inputs that appear legitimate to both humans and machines but are engineered to cause incorrect predictions by the target model [28]. These adversarial examples challenge the robustness of ML models, revealing vulnerabilities that may not be apparent through traditional testing methods. By using these AdvGAN-generated adversarial examples to train models, researchers can expose the models to a wider range of possible attacks. This makes the models better able to handle adversarial threats in the real world and improves model robustness against other attack vectors.

As the field of ML and sensor fusion continues to evolve, the use of GANs, particularly in adversarial settings, exemplifies the dynamic interplay between attack and defence. Each advancement in attack methodologies, such as those driven by AdvGAN, prompts the development of more sophisticated defences, pushing the boundaries of what these systems can achieve. This continuous cycle of innovation not only strengthens the resilience of ML systems but also deepens our understanding of their underlying mechanisms, ultimately contributing to the creation of more secure and reliable technologies.

2.3. Some Challenges Faced by Sensor Fusion Systems

Sensor fusion systems integrate data from multiple sources to achieve more accurate, reliable, and comprehensive understanding of the environment. These systems are widely used in applications such as autonomous vehicles, robotics, and surveillance. However, despite the significant benefits, sensor fusion systems face several challenges. These challenges arise from the complexity of merging data from different sensors, each with varying noise levels, reliability, and resolution.

Factors such as sensor failure, misalignment, and adversarial attacks can degrade the system's performance, leading to inaccurate or misleading outputs. Understanding and mitigating these challenges is crucial to ensuring the robustness and reliability of sensor fusion technologies. In this section, we will explore specific challenges such as the impact of perturbations on Kalman Filter-based fusion systems and fusion of data using Machine Learning methods.

2.3.1. Impact of Perturbations on Kalman Filter-Based Systems

Kalman Filters are a fundamental component of many sensor fusion systems, known for their ability to integrate noisy sensor data and provide optimal estimates of the system's state. However, these systems are vulnerable to adversarial perturbations—deliberate modifications to sensor data that can lead to inaccurate state estimates. In scenarios where precision is critical, such as autonomous systems, even small perturbations can cause significant deviations in the predicted state, which may result in unsafe decisions. This subsection discusses the mechanics of Kalman Filter-based systems, the influence of perturbations, and how adversarial attacks can exploit vulnerabilities to compromise system accuracy and reliability.

Introduction to Kalman Filters

The Kalman Filter operates through a repetitive two-step process: prediction and update. These steps allow the filter to process all available measurements sequentially, refining the estimation of the state vector with each new measurement. The implementation of Kalman filters in multi-sensor fusion systems significantly enhances the precision and reliability of the fused data [38, 5]. It reduces the impact of sensor noise and compensates for the intermittent availability of data from different sensors [39].

The Kalman Filter estimates the state $\mathbf{x}_k$ of a system at time step k based on a previous state $\mathbf{x}_{k-1}$ and measurements from multiple sensors. Initially, the predicted state $\hat{\mathbf{x}}_k^-$ is computed using the previous state and the control inputs through the state transition matrix $\mathbf{A}$ and control input matrix $\mathbf{B}$, respectively. Alongside this, the predicted error covariance $\mathbf{P}_k^-$ is updated based on the transition matrices and process noise covariance $\mathbf{Q}$.

During the update phase, the Kalman Filter incorporates new measurements $\mathbf{z}_k$ from multiple sensors. Each sensor may have different measurement models represented by the measurement matrix $\mathbf{H}$, and measurement noise characterized by the covariance matrix $\mathbf{R}$. These measurements adjust the predicted state to the updated state $\hat{\mathbf{x}}_k$ through the Kalman Gain $\mathbf{K}_k$.

$$
\begin{aligned}
\hat{\mathbf{x}}_k^- &= \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \\
\mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \\
\mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \\
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \\
\mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-
\end{aligned}
$$

Example of an Adversarial Attack

Consider an autonomous system that uses sensor fusion to integrate data from multiple sensors such as LiDAR, radar, and cameras. The system uses a Kalman Filter (KF) to fuse data from these sensors for accurate state estimation. An adversary introduces small perturbations to the LiDAR measurements, which propagate through the Kalman Filter, causing inaccurate state estimations.

Suppose the system tracks the position and velocity of an object in two dimensions, meaning the state vector is $\mathbf{x}_k = [x, y, v_x, v_y]^T$. If the adversary perturbs the LiDAR measurements $\mathbf{z}_k$ by adding a small noise $\boldsymbol{\delta}_k$, the perturbed measurement becomes:

$$\mathbf{z}'_k = \mathbf{z}_k + \boldsymbol{\delta}_k$$

The updated state estimate with the perturbed measurement is:

$$\begin{aligned}
\hat{\mathbf{x}}'_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}'_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \\
&= \hat{\mathbf{x}}_k^- + \mathbf{K}_k((\mathbf{z}_k + \boldsymbol{\delta}_k) - \mathbf{H}\hat{\mathbf{x}}_k^-) \\
&= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^- + \boldsymbol{\delta}_k) \\
&= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) + \mathbf{K}_k\boldsymbol{\delta}_k \\
&= \hat{\mathbf{x}}_k + \mathbf{K}_k\boldsymbol{\delta}_k
\end{aligned}$$

Here, $\mathbf{K}_k\boldsymbol{\delta}_k$ represents the deviation in the state estimate due to the perturbation $\boldsymbol{\delta}_k$. Even if $\boldsymbol{\delta}_k$ is small, it can cause significant errors depending on the Kalman Gain $\mathbf{K}_k$.

Consider a two-dimensional state (e.g., the x- and y-coordinates of the moving subject). Assume:

$$\mathbf{A} = \mathbf{I}_{2\times 2}, \quad \mathbf{H} = \mathbf{I}_{2\times 2} \quad \text{(Direct position observation)}$$

$$\mathbf{Q} = 0{,}1 \cdot \mathbf{I}_{2\times 2} \quad \text{(Process noise covariance)}$$

$$\mathbf{R} = 0{,}1 \cdot \mathbf{I}_{2\times 2} \quad \text{(Measurement noise covariance)}$$

Let the initial state estimate $\hat{\mathbf{x}}_0 = [0, 0]^T$ and the error covariance $\mathbf{P}_0 = \mathbf{I}$.
At time step $k = 1$, for one sensor:

- Prediction: $\hat{\mathbf{x}}_1^- = \hat{\mathbf{x}}_0 = [0, 0]^T$
- $\mathbf{P}_1^- = \mathbf{P}_0 + \mathbf{Q} = \mathbf{I} + 0{,}1\mathbf{I} = 1{,}1 \cdot \mathbf{I}$

Measurement $\mathbf{z}_1 = [1, 1]^T$, perturbed by $\boldsymbol{\delta}_1 = [0{,}1, 0{,}2]^T$, so $\mathbf{z}'_1 = [1{,}1, 1{,}2]^T$.

- Kalman Gain: $\mathbf{K}_1 = \frac{1{,}1}{1{,}1+0{,}1} = 0{,}917 \cdot \mathbf{I}$
- Updated state estimate: $\hat{\mathbf{x}}_1 = [0, 0]^T + 0{,}917 \cdot ([1, 1]^T - [0, 0]^T) = [0{,}917, 0{,}917]^T$

With perturbation:

- Updated state estimate: $\hat{\mathbf{x}}'_1 = [0, 0]^T + 0{,}917 \cdot ([1{,}1, 1{,}2]^T - [0, 0]^T) = [1{,}009, 1{,}100]^T$

The state estimate deviation due to the attack is $\hat{\mathbf{x}}_1' - \hat{\mathbf{x}}_1 = [1{,}009, 1{,}100]^T - [0{,}917, 0{,}917]^T = [0{,}092, 0{,}183]^T$.

This example shows how even small perturbations introduced in one sensor (LiDAR in this case) can propagate and cause significant deviations in a multi-sensor Kalman Filter-based system. The Kalman Gain, which amplifies the effect of the measurement error, plays a crucial role in the magnitude of the perturbation's impact. A coordinated attack that perturbs more sensors can lead to large cumulative errors, potentially causing incorrect decisions.

2.3.2. Catastrophic Fusion

Data Fusion doesn't always lead to more comprehensive output. It can lead to **catastrophic fusion** [2, 40]. As discussed, the primary goal is to enhance system performance by reducing uncertainty and noise; in some cases, combining sensor data may lead to worse outcomes—a phenomenon known as catastrophic fusion. This degradation in performance can be due to factors such as incompatible sensor modalities, poor calibration, or misalignment of sensor data [41]. While sensor fusion is designed to enhance measurement accuracy and reliability, it can degrade performance under certain conditions.

Assume two sensors, Sensor A and Sensor B, measure the same physical parameter with inherent noise. The measurements and noises are modelled as follows:

- Let x represent the true value of the measured parameter.
- The measurements from Sensor A and Sensor B are given by $z_A = x + n_A$ and $z_B = x + n_B$, respectively, where n_A and n_B are the noise components associated with each sensor.

Assuming n_A and n_B are Gaussian noises with zero mean and variances σ_A^2 and σ_B^2 respectively, the noise characteristics are as follows.

When fusing the measurements from the two sensors by simple averaging, the combined measurement is:

$$z_{\text{fused}} = \frac{z_A + z_B}{2} = x + \frac{n_A + n_B}{2}$$

If n_A and n_B are independent, the variance of the fused measurement is calculated as:

$$\text{Var}(z_{\text{fused}}) = \text{Var}\left(\frac{n_A + n_B}{2}\right) = \frac{1}{4}(\sigma_A^2 + \sigma_B^2)$$

Catastrophic fusion can occur under several conditions [40]:

1. **High Correlation Between Noises:** If the noises n_A and n_B are highly correlated, the expected benefit of noise reduction through averaging may not materialise, potentially leading to increased overall noise.

2. **Calibration Errors:** Misalignment or calibration errors can result in systematic biases that are amplified rather than mitigated when data from multiple sensors are combined.

3. **Disproportionate Noise Levels:** If one sensor has significantly higher noise variance than another, averaging their outputs can result in a fused measurement that is less accurate than the output of the less noisy sensor alone.

Motivation: This catastrophic fusion is exactly what an attacker can also exploit in ML systems. Depending on the target model it may be too confident on it's prediction. The issue here is of having **Redundant Data Sources**, where multiple sensors provide similar data, the fusion process can amplify the effect of noise or adversarial perturbations rather than mitigate it. This redundancy can lead to overfitting or a false sense of confidence in the fused output, particularly in localised environments where sensor inputs are highly correlated. This especially happens in cases where sensors agree due to shared vulnerabilities (e.g., all sensors being similarly fooled by an adversarial example, see chapter 3), the fusion process may mistakenly interpret this consensus as increased reliability, leading to catastrophic decisions. Identifying and mitigating these conditions is crucial for effectively applying sensor fusion technologies.

2.3.3. Lack of Interpretability and Transparency in ML Systems

ML-based systems using deep learning models, inherently operate as black boxes [42], meaning their decision-making processes are not easily interpretable. These models learn complex patterns from the data and make predictions based on these learned patterns. This lack of interpretability it challenging to diagnose or correct erroneous behaviour. Techniques such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are prominent examples of tools designed to provide insights into the decision-making processes of complex models [43].

If a sensor fusion system fails or produces an unexpected result, understanding the cause is crucial for correcting the issue. In traditional systems, each component (like a Kalman filter) is well-understood, and its outputs can be traced and analysed. In contrast, ML models do not provide clear explanations for their decisions, making it difficult to identify whether the failure is due to a sensor anomaly, a data preprocessing error, or an inherent flaw in the model's learning process [44]. So, adversarial training is a crucial one of the few ways to address these examples 3.1.2.

ML-based systems can struggle when they encounter data that is significantly different from what they were trained on (known as out-of-distribution data). Traditional sensor fusion systems are often designed with specific rules to handle unexpected inputs, but ML models might extrapolate in unpredictable ways, leading to unreliable outputs [45] . This unpredictability can be exploited by adversaries who craft inputs that fall outside the training distribution but may still appear valid to the system.

These limitations collectively make ML-based sensor fusion systems more prone to adversarial attacks, as attackers can exploit the lack of transparency that are

unlikely to be detected or understood by the model's operators. Addressing these limitations requires developing more resistant models, creating better diagnostic tools, and ensuring rigorous testing and validation processes are in place before deployment.

2.3.4. Limited Generalisation of Existing Defence Methods

Cross-Modality Vulnerabilities

A significant challenge related to existing defence methods is the potential for cross-modality vulnerabilities. In a multi-sensor fusion system, different sensors provide complementary data that, when combined, should ideally enhance the system's overall accuracy and reliability. However as seen, adversarial attacks can exploit the interactions between these modalities in unexpected ways, potentially leading to system-wide failures.

For instance, an adversarial perturbation introduced into the data stream of one sensor modality (e.g., a slight distortion in LiDAR data) could propagate through the fusion process, resulting in erroneous outputs from the entire system. This issue becomes more complex when different sensor modalities are tightly coupled within the fusion process, as an attack on one modality could inadvertently affect others, even if they were not directly targeted.

Let's consider how an adversarial perturbation $\delta_i(t)$ in one sensor's data stream can propagate through the fusion process, affecting the entire system's output. From section 2.1, the fusion process combines the inputs from all sensors into a final output.

$$\Psi : z_f(t) = F(\Phi_1, \Phi_2, \Phi_3, \ldots, \Phi_n) \tag{1}$$

Say an adversary $\delta_1(t)$ is introduced in the data stream of sensor S_1, the perturbed measurement becomes:

$$z_1'(t) = z_1(t) + \delta_1(t) \tag{2}$$

From equation 1 and 2, the fusion process combines the inputs from all sensors into a final output $\Psi'(t)$:

$$\Psi'(t) = F\left(\Phi_1(z_1'(t)), \Phi_2(z_2(t)), \ldots, \Phi_n(z_n(t))\right) \tag{3}$$

First-order Taylor series expansion:

$$\Psi'(t) \approx \Psi(t) + \sum_{i=1}^{n} \frac{\partial F}{\partial \Phi_i(z_i(t))} \cdot \frac{\partial \Phi_i}{\partial z_i(t)} \cdot \delta_i(t)$$

For the case where the perturbation is only in S_1:

$$\Psi'(t) \approx \Psi(t) + \frac{\partial F}{\partial \Phi_1(z_1(t))} \cdot \frac{\partial \Phi_1}{\partial z_1(t)} \cdot \delta_1(t)$$

This equation shows that the perturbation $\delta_1(t)$ in the data stream of S_1 propagates through the fusion process, influencing the entire system's output $\Psi'(t)$.

When different sensor modalities are tightly coupled, the effect of $\delta_1(t)$ on $\Psi'(t)$ can be non-trivial, particularly if the derivative $\frac{\partial F}{\partial \Phi_1(z_1(t))}$ is large, indicating a strong dependency of the fusion output on S_1. Moreover, if there are interdependencies between sensors (e.g., $\Phi_i(z_i(t))$ depends on $z_1(t)$), the perturbation in S_1 could indirectly affect the processing of other sensors $S_2, S_3, \ldots,$. This is usually the case if the fusion step is performed at intermediate or late stages.

Let's look at why this might be the case even if the input from all sensors is equally weighted. For e.g. in early fusion, assuming all sensors nominally equal, the impact is scaled by $\frac{1}{n}$, see 2.4.1, suggesting that the effect of the perturbation is less (spread out) due to the equal reliance on all sensors. The system output in this case is:

$$\Psi'(t) \approx \Psi(t) + \frac{1}{n} \cdot \frac{\partial F}{\partial \Psi(t)} \cdot \frac{\partial \Phi_1(z_1(t))}{\partial z_1(t)} \cdot \delta_1(t)$$

However, even with equal weighting, the perturbation $\delta_1(t)$ can still cause significant disruption if $\frac{\partial F}{\partial \Psi(t)}$ is highly sensitive to changes in any of its inputs. If the perturbation $\delta_1(t)$ is crafted to exploit specific vulnerabilities in $\Phi_1(z_1(t))$ or in the fusion process F, it can cause a ripple effect, leading to a disproportionately large impact on $\Psi'(t)$ despite the normalised design.

Example of an attack targeting cross-modality vulnerabilities

Consider a scenario where G is a multi-layered neural network that processes concatenated sensor data to **classify** an action played by a **human subject**. The perturbation $\delta_1(t)$ introduced in the measurement of S_1 could shift the input $Z'(t)$ in a way that causes the network to misclassify an action.

Linear Case

If G is a linear function or a simple model (e.g., a linear classifier), the effect of the perturbation $\delta_1(t)$ might be diluted across the concatenated input vector. This is where the strength of sensor-fusion comes to rescue and helps defend an attack. Though, as discussed earlier, there are scenarios, for e.g. catastrophic fusion 2.3.2, where it can still lead to misclassifications.

Non-linear Case

In practice, G is often nonlinear (e.g., a neural network). Nonlinear models can have regions where small perturbations in the input space lead to large changes in the output. In this case, even though all sensor data is fused early, a perturbation in one sensor can push the input $Z'(t)$ into a region of the model's input space where the output $\Psi'(t)$ changes drastically, leading to a large error or misclassification.

Even if the model G is designed to be equally sensitive to all sensors (i.e., no sensor dominates the decision), the adversarial perturbation $\delta_1(t)$ can still have a large impact if it moves the combined input $Z'(t)$ into a highly nonlinear region of the model's

decision boundary. While the impact might seem less pronounced than in intermediate (where interdependencies between processed data from different sensors can amplify the effect), the nonlinearity and sensitivity of the model to specific input dimensions can still result in a significant disruption of the system's output. Therefore, early fusion is not inherently safer from such attacks—it just exhibits different dynamics that still require robust defences. With late fusion, the final decision is based on a combination of predictions, making it harder for an attacker to craft a perturbation that misleads all models simultaneously.

Motivation: Existing defence strategies often do not account for these cross-modality interactions, focusing instead on protecting individual sensors in isolation [24]. By exploring how adversarial attacks on one sensor can influence the fusion process and affect the data from other sensors, it becomes possible to propose defence mechanisms that are more holistic, protecting the system as a whole rather than just individual components.

Specificity of Attacks and Defences

A major challenge in sensor fusion, especially when integrated with machine learning, is the limited generalisation of existing methods for adversarial attacks and defences. Many studies tend to focus on specific types of sensors or particular attack vectors, leading to solutions that are not broadly applicable across different systems or scenarios. This narrow focus poses significant limitations when these systems are deployed in real-world applications, where the range of possible attacks and environmental conditions is much more diverse.

For example, research might emphasise the impact of adversarial attacks on image-based sensors like RGB cameras, with defence mechanisms specifically designed for these threats. However, such approaches often do not extend effectively to other sensor types, such as LiDAR, radar, or inertial measurement units (IMUs), each of which might be part of a comprehensive sensor fusion system. Vulnerabilities and defences relevant to one type of sensor do not necessarily apply to others, leaving parts of the system exposed to attacks that current defences cannot mitigate.

Motivation: The research should include a wider range of sensors and attack types for the development of more versatile and generalised defence strategies. By evaluating the impact of adversarial attacks across multiple sensor modalities, it is possible to gain insights into how these attacks can affect the entire sensor fusion system. This comprehensive approach enables the development of defence mechanisms that are not only effective against specific types of attacks but are also robust across different sensor types and configurations.

2.4. Sensor Uncertainty and Fusion Models

One of the critical challenges in sensor fusion is managing the inherent uncertainty and noise in sensor data, which can compromise the reliability of the system [46]. Sensor inaccuracies, environmental interference, and signal degradation are common factors that introduce uncertainty. To address these issues, sensor fusion systems often

employ sophisticated modelling techniques to estimate and incorporate uncertainty into the fusion process. Machine learning models, with their capacity to learn from data and improve over time, are particularly effective in modelling and mitigating these uncertainties. Modern systems use probabilistic models and techniques, such as Bayesian estimation, to account for sensor errors and stochastic environmental influences. These models describe the likelihood of various outcomes based on sensor data, which is crucial for interpreting readings in noisy, incomplete, or uncertain conditions.

This uncertainty and noise in sensor data also present significant vulnerabilities that adversarial attacks can exploit [47]. Attackers can manipulate input data to disrupt the fusion process, leading to inaccurate or malicious state estimations. For example, uncertainty in sensor readings—whether due to noise, environmental factors, or sensor malfunctions—can be deliberately amplified or distorted by attackers to degrade system performance. This underscores the importance of robust uncertainty modelling as a defence mechanism. By effectively quantifying and incorporating uncertainty into sensor fusion models, systems not only improve the accuracy and reliability of state estimates but also enhance their resilience against potential adversarial exploits. Therefore, robust uncertainty modelling is fundamental in safeguarding sensor fusion processes, ensuring both accuracy and security in environments where data integrity is paramount.

2.4.1. Fusion Types

Noise Reduction through Fusion Across Same Sensors

Fusing data from multiple instances of the same type of sensor can reduce noise through statistical averaging. The measurements from sensors $S_1, S_2, \ldots, S_n$ observing the same quantity x with independent additive noise ϵ_i are given by:

$$y_i = x + \epsilon_i$$

The average of these measurements is:

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i = x + \frac{1}{n}\sum_{i=1}^{n} \epsilon_i$$

Assuming the noise ϵ_i has zero mean and variance σ^2, the variance of the average noise term reduces to $\frac{\sigma^2}{n}$.

Noise Reduction Through Different Sensor Types (e.g., Position and Velocity)

For sensors measuring position x and velocity v, the state estimation problem can be formulated as follows, with state transition and measurement equations:

$$\begin{bmatrix} x \\ v \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}_{k} + \text{noise}$$

$$y_x = x + \text{noise}, \quad y_v = v + \text{noise}$$

The state estimation involves predicting and updating the state $\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix}$ using the Kalman Filter. The Kalman Filter update equations refine the state estimate $\hat{\mathbf{x}}_{k|k}$ and error covariance $\mathbf{P}_{k|k}$ after each measurement. The Kalman gain $\mathbf{K}_k$ adjusts the predicted state $\hat{\mathbf{x}}_{k|k-1}$ using the difference between the actual measurements $\mathbf{y}_k$ and the predicted measurements $\mathbf{H}\hat{\mathbf{x}}_{k|k-1}$. The updated error covariance $\mathbf{P}_{k|k}$ reflects reduced uncertainty, as indicated by $\mathbf{P}_{k|k} < \mathbf{P}_{k|k-1}$, meaning the estimation error variance decreases with each update, see 2.3.1.

$$\begin{aligned} K_k &= P_k H^T (H P_k H^T + R)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - H\hat{x}_{k|k-1}) \\ P_{k|k} &= (I - K_k H) P_{k|k-1} \end{aligned}$$

The error covariance update shows that $P_{k|k} < P_{k|k-1}$, indicating a reduction in the estimation error variance post-update.

Fusion Across Time

Using historical data for current state estimation:

$$\begin{aligned} \hat{x}_{k|n} &= \hat{x}_{k|k} + A_k(\hat{x}_{k+1|n} - \hat{x}_{k+1|k}) \\ P_{k|n} &= P_{k|k} + A_k(P_{k+1|n} - P_{k+1|k})A_k^T \end{aligned}$$

where $A_k = P_{k|k}K_{k+1}^T P_{k+1|k}^{-1}$ is the smoothing gain. The update to the error covariance matrix $P_{k|n}$ shows that the error variance in the smoothed estimate is less than that of the filtered estimate, thus reducing the overall noise and improving the estimation accuracy.

2.4.2. State Estimation Techniques in Sensor Fusion

In machine learning-based sensor fusion, the goal is to combine information from multiple sensors to improve the estimation of the state $\mathbf{x}$. Each sensor provides a measurement $\mathbf{z}_i$, which can be modelled as:

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}) + \mathbf{v}_i$$

where:

- $\mathbf{h}_i(\mathbf{x})$ is the measurement function for sensor i,
- $\mathbf{v}_i$ is the measurement noise, assumed to be Gaussian with zero mean and covariance $\mathbf{R}_i$.

The objective in sensor fusion is to find the best estimate of the state x given all the measurements from the sensors. Bayesian estimation allows us to update our knowledge about the state x using new data (sensor measurements). The derivation to getting the Posterior distribution is provided in Appendix 7.1.

- **Prior Distribution** $p(x)$: Initial belief about x.

- **Likelihood** $p(z_1, z_2, \ldots, z_n \mid x)$: Probability of measurements given x.
- **Posterior Distribution** $p(x \mid z_1, z_2, \ldots, z_n)$: Updated belief about x after considering the measurements.

The MAP estimate maximises the posterior distribution:

$$\hat{x}_{\text{MAP}} = \arg\max_x p(x \mid z_1, z_2, \ldots, z_n)$$

For Gaussian noise, the posterior is also Gaussian, making the MAP estimate the mean of the posterior. Key methods for state estimation include but not limited to:

- **Extended Kalman Filter (EKF)**: Linearizes non-linear measurement functions and applies the Kalman filter.
- **Particle Filter**: Uses particles to approximate the posterior for highly non-linear systems.
- **Neural Networks**: ML models, particularly deep learning, can learn complex, non-linear mappings between sensor measurements and state estimates, offering a flexible alternative for challenging estimation problems.

Neural networks (NNs) can be trained to model complex relationships between multiple sensor inputs and the underlying state. For example, consider a simple feed-forward neural network (the information only flows in one direction, from the input layer through any hidden layers to the output layer), that takes measurements from multiple sensors and produces an estimate of the state $\hat{x}$:

$$\hat{x} = f(z_1, z_2, \ldots, z_n; \theta) \tag{4}$$

where $f(\cdot)$ is the neural network function parameterised by θ, and z_i are the sensor measurements.

The training process minimises a loss function $L(\hat{x}, x)$, such as the mean squared error, see 3.3.2, between the predicted state $\hat{x}$ and the true state x:

$$\theta^* = \arg\min_\theta L(\hat{x}, x)$$

This approach enables the NN to learn the optimal fusion strategy directly from data, capturing non-linearities that traditional linear models like Kalman filters cannot.

Enhanced State Estimation Through Multi-Sensor Data Fusion

To demonstrate the advantage of combining data from multiple sensors, consider the following two scenarios: using a single sensor to estimate the state $\mathbf{x}$, and using multiple sensors and fusing their data to estimate the state $\mathbf{x}$.

Single Sensor Case

For a single sensor providing measurement $\mathbf{z}_1$, the estimation error is given by:

$$\mathbf{e}_1 = \mathbf{x} - \hat{\mathbf{x}}_1$$

where $\hat{\mathbf{x}}_1$ is the state estimate based on $\mathbf{z}_1$. Assuming Gaussian noise with covariance $\mathbf{R}_1$, the estimation error covariance is:

$$\mathbf{P}_1 = \mathbf{H}_1 \mathbf{R}_1 \mathbf{H}_1^T$$

Multiple Sensors Case

For multiple sensors providing measurements $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$, the combined state estimate $\hat{\mathbf{x}}$ is obtained by fusing the individual measurements. The key advantage of this approach is reflected in the estimation error covariance, see Appendix 7.2:

$$\mathbf{P} = \left(\sum_{i=1}^{n} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i \right)^{-1}$$

$$\mathbf{P} = \left(\mathbf{H}_1^T \mathbf{R}_1^{-1} \mathbf{H}_1 + \mathbf{H}_2^T \mathbf{R}_2^{-1} \mathbf{H}_2 \right)^{-1} \text{(For } S1 \text{ and } S2\text{)}$$

This expression shows that the fused estimation error covariance $\mathbf{P}$ is typically lower than that of any individual sensor as $\mathbf{P} \leq \mathbf{P_1}$ and $\mathbf{P} \leq \mathbf{P_2}$.

2.5. Neural Networks Models for Sensor Fusion

Neural networks can be utilised to model the complex relationships between multiple sensor inputs and the state estimate. Let $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n]$ represent the concatenated sensor measurements. This approach is an example of early fusion, where all sensor inputs are concatenated and processed together at the input layer. However, the same or a similar concept can be applied to intermediate or late fusion by adjusting where and how the sensor data is combined within the network architecture, see 2.6.2 Fusion Architectures. A neural network can be trained to approximate the mapping $f : \mathbf{z} \to \hat{\mathbf{x}}$.

Consider the same feed-forward NN, equation 4, with L layers. The output of each layer l is given by:

$$\mathbf{h}^{(l)} = \sigma \left(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \right) \tag{5}$$

Where:

- $\mathbf{h}^{(0)} = \mathbf{z}$ is the input layer,
- $\mathbf{h}^{(L)} = \hat{\mathbf{x}}$ is the output layer,
- $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector for layer l,

- σ is the activation function (e.g., ReLU, sigmoid)

Activation functions, like ReLU and sigmoid, introduce non-linearity into neural networks, allowing them to model complex patterns and make decisions by transforming the input signals of neurons, see 2.6.3. The network parameters $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$ are learned by minimising a loss function that measures the difference between the predicted state $\hat{\mathbf{x}}$ and the true state $\mathbf{x}$. For example, using mean squared error (MSE):

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\|^2$$

Where N is the number of training samples.
The training process involves:

1. Forward propagation: Compute the network output $\hat{\mathbf{x}}$ for each training sample.
2. Loss calculation: Compute the loss $\mathcal{L}$.
3. Backward propagation: Compute the gradients of the loss with respect to the network parameters.
4. Parameter update: Update the network parameters using an optimisation algorithm like gradient descent.

The neural network performs sensor fusion by learning the weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$ that best combine the sensor measurements to estimate the state $\mathbf{x}$. Mathematically, the fusion process can be represented as:

$$\hat{\mathbf{x}} = f(\mathbf{z}; \theta)$$

where $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$ represents the network parameters.

2.6. Machine Learning for Sensor Data Interpretation

Machine learning algorithms are adept at interpreting sensor data, extracting meaningful patterns, and making predictions. For example, convolutional neural networks (CNNs) are widely used in processing and interpreting visual data from cameras, while recurrent neural networks (RNNs) effectively handle time-series data from motion sensors. Integrating ML models into sensor fusion systems enables more intelligent and automated decision-making based on a comprehensive sensor data analysis.

2.6.1. An Overview of Machine Learning Algorithms and Their Susceptibility to Adversarial Attacks

This section provides an overview of some key machine learning algorithms, including Neural Networks (NNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Support Vector Machines (SVMs), highlighting their functionalities, common applications, and susceptibility to adversarial attacks. The discussion also covers some existing strategies for mitigating these vulnerabilities, ensuring more robust and reliable models in practice.

Neural Networks (NNs)

Neural networks consist of layers of interconnected nodes. Each connection has a weight that is adjusted during the training process [48]. Deep Learning involves NNs with multiple hidden layers, allowing the model to learn complex patterns in data [49]. Neural Networks are used for their ability to handle high-dimensional data and learn complex patterns. Convolutional Neural Networks (CNNs) are particularly popular for sensor image and video data [50].

NNs are prone to adversarial attacks that make subtle alterations to input data [34]. Common attacks include Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). Defensive techniques like adversarial training, where models are trained on adversarial examples, and defensive distillation, are used to increase robustness. Bayesian Neural Networks, which provide better uncertainty estimates, can also help in detecting adversarial inputs [51], see 2.4.2.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

RNNs are designed to recognise patterns in data sequences, such as time series or text. LSTMs are a special kind of RNN capable of learning long-term dependencies [52]. RNNs process sequences by maintaining a 'memory' of previous inputs [53]. LSTMs are a special kind of RNN that are better at capturing long-term dependencies, using structures called gates. Ideal for time-series data, RNNs and LSTMs are used where the sequence of data is important, such as in motion tracking or speech recognition.

RNNs and LSTMs are susceptible to adversarial attacks that target critical points in the data sequence [54], potentially disrupting the memory of the model and leading to incorrect outputs. These attacks often exploit the sequential nature of the data. This is especially dangerous in real-time applications such as autonomous driving. Employing temporal consistency checks, which ensure logical predictions over time, can help mitigate these attacks. Additionally, using ensembles of RNNs and LSTMs can make it harder for adversarial examples to deceive multiple models simultaneously [55].

Support Vector Machines (SVMs)

SVMs are a set of supervised learning methods used for classification, regression, and outliers detection [56]. The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. Effective for

classification and regression tasks, SVMs are used in sensor fusion for pattern recognition, such as identifying objects or anomalies in sensor data.

Although SVMs are more robust to certain types of noise compared to neural networks, they are still vulnerable to adversarial attacks. An attacker can manipulate input data to shift the decision boundary, leading to misclassification. To defend against these attacks, techniques such as outlier detection and robust SVM variants that are less sensitive to data perturbations can be employed [57]. Additionally, adding noise to the training data, similar to adversarial training, can enhance the model's resilience to attacks [58].

2.6.2. Fusion Architectures

Early Fusion:

In early fusion, sensor data is concatenated and fed into the neural network as a single input. Mathematically, if $z_1, z_2, \ldots, z_n$ are the sensor inputs, the concatenated input z is:

$$z = [z_1, z_2, \ldots, z_n]$$

This combined input z is processed by the network to estimate the state $\hat{x}$:

$$\hat{x} = \text{NN}(z)$$

Intermediate Fusion:

Intermediate fusion involves processing each sensor input individually through separate neural network branches before combining their intermediate representations:

$$h_1^{(l)} = \sigma(W_1^{(l)} h_1^{(l-1)} + b_1^{(l)})$$

$$h_2^{(l)} = \sigma(W_2^{(l)} h_2^{(l-1)} + b_2^{(l)})$$

The intermediate representations are then concatenated or combined in another manner before further processing:

$$h_{\text{fused}} = \text{Combine}(h_1^{(L_1)}, h_2^{(L_2)}, \ldots, h_n^{(L_n)})$$

$$\hat{x} = \text{NN}(h_{\text{fused}})$$

Late Fusion:

Late fusion processes each sensor input through separate neural network branches to the end and then combines their outputs.

$$\hat{x}^1 = \text{NN}_1(z_1)$$

$$\hat{x}^2 = \text{NN}_2(z_2)$$

The final estimate is a weighted combination of these outputs:

$$\hat{x} = \sum_{i=1}^{n} w_i \hat{x}^i$$

Multimodal Learning

Multimodal learning is defined when different types of sensor data (e.g., RGB, IR, motion) are processed jointly to improve the learning process. Fusion can occur at various stages in the learning pipeline and the stages can be different for different sensors (hybrid). For e.g.

$$z_{\text{RGB}}, z_{\text{IR}}, z_{\text{motion}} \rightarrow \text{NN}_{\text{RGB + IR}}, \text{NN}_{\text{motion}}$$

2.6.3. System Architecture for a Human-Action Recognition Model

Let's take a system that contains multiple sensors for human action recognition in a room. An implemented fused CNN employs **intermediate** (feature-level) fusion with a combination of RGB, IR, and motion data. Each data stream is processed through separate CNN branches that extract relevant features before fusion.

Convolutional Layer: Each branch initiates with a convolutional layer that applies multiple filters to detect essential features, such as edges, textures, and movement patterns inherent to each modality.

Batch Normalisation: To ensure stable and accelerated training, the outputs of the convolutional layers are normalised. This process adjusts and scales the activations, mitigating internal covariate shifts.

ReLU Activation: A ReLU (Rectified Linear Unit) function introduces non-linearity into the model, allowing it to capture complex patterns and interactions within the data.

Max Pooling: Following activation, max pooling layers reduce the spatial dimensions of the feature maps, retaining the most salient features while reducing computational load.

Softmax: The softmax function converts raw output scores (logits) into probabilities, normalising them across classes so that each value falls between 0 and 1, and the sum equals 1.

From equation 5, each modality-specific CNN branch processes its input to produce an intermediate feature representation. The output of each layer in the fusion process is mathematically represented as:

$$\begin{aligned}
\mathbf{h}_{\text{RGB}}^{(1)} &= \text{ReLU}(\text{BatchNorm}(\mathbf{W}_{\text{RGB}}^{(1)} * \mathbf{z}_{\text{RGB}} + \mathbf{b}_{\text{RGB}}^{(1)})) \\
\mathbf{h}_{\text{IR}}^{(1)} &= \text{ReLU}(\text{BatchNorm}(\mathbf{W}_{\text{IR}}^{(1)} * \mathbf{z}_{\text{IR}} + \mathbf{b}_{\text{IR}}^{(1)})) \\
\mathbf{h}_{\text{motion}}^{(1)} &= \text{ReLU}(\text{BatchNorm}(\mathbf{W}_{\text{motion}}^{(1)} * \mathbf{z}_{\text{motion}} + \mathbf{b}_{\text{motion}}^{(1)}))
\end{aligned}$$

***** : Convolution operation

Fusion Layer

The features from the RGB, IR, and motion streams are concatenated:

$$\mathbf{h}_{\text{fused}} = [\mathbf{h}_{\text{RGB}}^{(L)}, \mathbf{h}_{\text{IR}}^{(L)}, \mathbf{h}_{\text{motion}}^{(L)}]$$

This fused representation is processed through fully connected layers to output the final action recognition result:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_{\text{fc}}\mathbf{h}_{\text{fused}} + \mathbf{b}_{\text{fc}})$$

The fused CNN is trained using a cross-entropy loss function, defined as:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

where y_i is the ground truth label and $\hat{y}_i$ is the predicted probability for the i-th class. The optimisation is performed using the Adam optimiser with a learning rate α:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}$$

Evaluation Metrics

The performance of the model is evaluated using accuracy, precision, recall, and F1-score:

- Accuracy: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Precision: Precision $= \frac{TP}{TP+FP}$
- Recall: Recall $= \frac{TP}{TP+FN}$
- F1-score: F1-score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. The quantitative results for three different fusion strategies, specific to the experiment conducted for this thesis, are summarised in Table 4. This table highlights the improvements achieved by the fused model predictions.

2.7. Applications and Challenges of Sensor Fusion

Sensor fusion systems are increasingly critical across various domains, including autonomous navigation, industrial automation, environmental monitoring, and particularly in security and surveillance [2]. However, as seen in this chapter, the complexity and interdependence of these systems also introduce vulnerabilities, particularly to adversarial attacks. The defence mechanisms must also account for the interactions between sensors and the potential for cross-sensor inconsistencies caused by an attack 2.3.4.

An attacker with knowledge of the system's architecture could subtly manipulate one or more sensor inputs to exploit the system's decision-making process [27]. Such vulnerabilities are not limited to physical sensors. In many systems, the fusion process

also includes data from digital sensors, such as network traffic monitors or GPS signals. An attacker could craft adversarial inputs that exploit the fusion algorithm itself, leading to a misclassification of the overall situation. For example, in a security operation where drones are used for perimeter surveillance, adversarial inputs could cause the drone to miscalculate its position or the location of potential intruders, leading to incorrect or delayed responses.

This added layer of complexity might make it easier for an attack to succeed or go undetected in a multi-sensor system compared to a single-sensor system. In a fusion system, an attack on one of the sensors can have a more significant impact than in a system relying on a single sensor. When only one sensor is used, standard defence mechanisms can often detect and mitigate the attack, preserving the accuracy of the estimation. However, in a multi-sensor fusion scenario, the same attack on the affected sensor can introduce conflicting data into the fusion process. This conflicting data can degrade the overall estimation, as the fusion system might erroneously weigh the corrupted sensor data more heavily, leading to an inaccurate or compromised result. Despite applying the same defences at both the sensor level and the fused output, these defences may fail to fully address the complexity introduced by multiple data sources, making the system more vulnerable in the multi-sensor case [59].

3. ADVERSARIAL ATTACKS AND DEFENCES

Adversarial attacks in machine learning (ML) refer to a class of techniques and strategies used to manipulate or deceive ML models by exploiting their vulnerabilities. These attacks are designed to cause ML models to make incorrect predictions or classifications, even when the input data appears almost identical to legitimate data. It involves crafting inputs to deceive machine learning models. These attacks can subtly alter inputs to cause misclassification, posing significant security concerns [60]. They can be Perturbation-based Attacks, Poisoning Attacks (altering training data) or Evasion Attacks. Understanding these attacks is crucial, especially in applications like autonomous vehicles and healthcare, where sensor fusion systems are vital. Adversarial attacks are particularly concerning because they can undermine the reliability and security of ML systems, especially in critical applications like security systems and healthcare [60].

3.1. Background and Related Work

3.1.1. Previous Research on Adversarial Attacks Methods

A substantial body of research has explored how perturbation-based attacks exploit vulnerabilities in ML models, especially deep learning algorithms. These attacks involve small, often imperceptible modifications to input data, which can lead to incorrect predictions or classifications by the model. Notably, Szegedy et al. [34] were among the first to reveal the existence of adversarial examples in deep neural networks.

Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al. [36], is one of the foundational perturbation-based attack techniques. FGSM generates adversarial examples by adding a perturbation to the input data, computed as the sign of the gradient of the loss function with respect to the input. This method has been extensively studied and applied across various domains, including sensor fusion systems in autonomous vehicles and robotics [61].

Research by Eykholt et al. [62] demonstrated that FGSM could effectively generate adversarial inputs across different sensor modalities, such as camera images and LiDAR data. Their experiments showed that FGSM-induced perturbations could lead to dangerous misclassifications in autonomous vehicles, such as the misidentification of traffic signs, thereby highlighting the vulnerability of sensor fusion systems to such attacks.

Basic Iterative Method (BIM)

The Basic Iterative Method (BIM), also known as Iterative FGSM (I-FGSM), was proposed by Kurakin et al. [21] as an extension of FGSM. BIM iteratively applies the FGSM attack with small step sizes, allowing for more precise perturbations and

increasing the likelihood of successful attacks. This method has been widely explored in the context of image classification and object detection systems, which are integral components of sensor fusion technologies.

Kurakin et al. [21] further investigated the impact of BIM on sensor fusion systems, demonstrating that iterative attacks could incrementally degrade the performance of individual sensors. Their work highlighted the cumulative effect of such attacks on the overall decision-making capability of the system, particularly in safety-critical applications like autonomous vehicles and medical diagnostics.

Projected Gradient Descent (PGD)

Projected Gradient Descent (PGD), as proposed by Madry et al. [63], builds on BIM by including a projection step that ensures the adversarial example remains within a predefined perturbation bound, making it a stronger and more effective attack.

Madry et al. [63] applied PGD to sensor fusion systems, showing that it systematically explores the most vulnerable regions of the input space. This attack was particularly effective in degrading the performance of sensor fusion systems used in security and surveillance applications, where the integration of data from multiple sensors is crucial for accurate threat detection.

PGD can be used to generate adversarial examples for multi-sensor surveillance systems combining visual and thermal data. The attack led to a significant increase in false alarms and missed detection's, severely compromising the system's reliability. This research highlights the necessity of developing robust defences against such sophisticated attacks.

Physical Attacks

While perturbation-based attacks primarily involve digital manipulations, recent research has focused on physical attacks that modify the environment to deceive ML models. These attacks pose a significant threat to sensor fusion systems, where inputs from various physical sensors, such as cameras, LiDAR, and radar, are combined to form a cohesive understanding of the environment. Early work by Athalye et al. [64] laid the foundation for studying the impact of physical adversarial examples on real-world systems, sparking interest in this area.

Adversarial patches, first explored by Brown et al. [65], represent a prominent type of physical attack. These patches are small, crafted patterns introduced into the environment to mislead ML models. Research has shown that these patches can cause significant misclassifications when placed on objects, the ground, or even worn by individuals [66].

Zhao et al. [67] investigated the impact of adversarial patches on sensor fusion systems in autonomous vehicles. Their study demonstrated that a patch placed on a stop sign could cause a camera-based system to misidentify it, while the accompanying LiDAR data still recognised it as an obstacle. This sensor mismatch can lead to incorrect decision-making, such as the failure to stop at intersections, thereby endangering road safety.

The work of Eykholt et al. [62] further illustrated the potential real-world impact of adversarial patches on autonomous vehicles. By placing small patches on road signs,

they successfully caused the vehicles' sensor fusion systems to misclassify these signs, leading to dangerous driving behaviours. This case study underscores the urgent need for developing countermeasures against physical adversarial attacks.

Model Poisoning and Backdoor Attacks

Model poisoning and backdoor attacks are sophisticated techniques that involve corrupting the training data or the training process to introduce vulnerabilities into the model. These attacks have been extensively studied in the context of ML systems, with recent work focusing on their implications for sensor fusion systems.

Gu et al. [68] provided an in-depth exploration of backdoor attacks, where a trigger pattern is inserted during training to cause the model to behave incorrectly when the trigger is present in the input. This approach has been shown to be particularly effective in scenarios where the model is deployed in a sensor fusion system, as the trigger can be embedded in the physical environment.

A study by Bagdasaryan et al. [69] examined the impact of model poisoning on sensor fusion systems, revealing how attackers could manipulate the training data for one sensor while leaving others intact. This creates discrepancies between sensor modalities during inference, leading to compromised system performance.

Recent literature has also highlighted emerging trends in adversarial attacks, such as the use of generative models to create realistic adversarial examples [37], and the exploration of adversarial attacks in federated learning and edge computing environments [69]. These trends indicate that the field is rapidly evolving, with new challenges continuously emerging. A study by Wu et al. [70] on multi-modal sensor systems in autonomous drones highlighted the dangers of combining physical adversarial patches with digital perturbations. The attack effectively bypassed the system's defences, leading to incorrect navigation decisions. This case study emphasises the need for ongoing research into advanced countermeasures to protect against such sophisticated threats.

3.1.2. Previous Research on Adversarial Training and Defences

Adversarial Training and Robust Optimisation

Adversarial training is a defence technique that involves training a machine learning model on a mixture of adversarial and clean examples. This approach aims to make the model robust against specific types of adversarial attacks by learning to correctly classify both perturbed and unperturbed inputs. Goodfellow et al. introduced one of the earliest forms of adversarial training, proposing the use of fast gradient sign method (FGSM) to generate adversarial examples during the training process [27]. The method applies a perturbation to the input data that maximises the loss of a model, intending to create more challenging data points for training.

While adversarial training has shown promise, it comes with significant limitations and challenges. One of the primary concerns is the computational cost associated with generating adversarial examples and retraining the model, which can be substantially higher than training on clean data alone. Furthermore, adversarial training tends

to reduce the model's accuracy on clean examples, a phenomenon known as the robustness-accuracy trade-off.

Another issue, as Carlini and Wagner noted, adversarially trained models under one norm type can remain vulnerable to attacks designed with a different perturbation strategy, indicating a lack of cross-norm robustness [71]. Which rises the issue of adaptive attacks, where attackers tailor their methods based on the defence mechanism, poses a continuous threat. Adversarially trained models might perform well against known attack strategies but fail against novel or modified attacks specifically designed to circumvent the training approach.

Detection-Based Defences

Anomaly detection in adversarial settings involves identifying inputs that deviate from the expected distribution of training data. Techniques such as statistical anomaly detection, which uses the statistical properties of the data, and machine learning-based methods, which learn a model of normal behaviour and detect deviations, are common. Hendrycks and Gimpel introduced a simple baseline for detecting adversarial examples by using softmax probabilities [72]. Models tend to produce less confident predictions for adversarial inputs, thus allowing for the potential detection of these inputs based on confidence thresholds.

Deep autoencoders have also been used for anomaly detection, where the autoencoder learns to reconstruct normal training data. During inference, significant reconstruction errors can indicate adversarial manipulations. This method has been effective in various contexts, especially where the normal operational profiles are well-defined and can be modelled accurately.

Input reconstruction strategies attempt to "sanitize" adversarial examples by altering them to resemble the clean data distribution before they are processed by the machine learning model. MagNet, proposed by Meng and Chen, uses a network of autoencoders that learn to reconstruct clean inputs from adversarial perturbations [73]. By comparing the reconstruction to the original input, the system can detect and mitigate small perturbations.

Another approach involves the use of Generative Adversarial Networks (GANs) to reconstruct inputs. Samangouei et al. introduced Defense-GAN, which leverages the generative capability of GANs to project perturbed inputs back onto the manifold of clean data [74]. This method has shown promise in defending against a range of static adversarial attacks, particularly in image processing tasks.

Hybrid Approaches and Cross-Modal Validation

Cross-modal validation exploits multiple data modalities (e.g., visual, auditory, textual) to check the consistency of the inputs and predictions. For sensor fusion systems, which integrate data from various sensors, cross-modal validation can serve as a critical layer of defence. By verifying the consistency of information across different sensory inputs, systems can detect and potentially reject inputs that are adversarial only in a single modality but not others.

Hybrid approaches to machine learning robustness combine multiple defence strategies to enhance overall system security. For example, combining adversarial

training with input reconstruction and anomaly detection can cover a broader range of attack vectors than any single method alone. Papernot et al. proposed a framework where multiple models, each trained with different defences, collaborate to make a final decision [75]. This ensemble approach not only improves robustness against a variety of attacks but also helps in mitigating the impact of the robustness-accuracy trade-off by diversifying the defence mechanisms.

Another method involves exploiting redundancy inherent in sensor fusion systems. Redundant systems, where multiple sensors provide overlapping data about the same physical property, can inherently defend against certain types of sensor-specific adversarial attacks. For example, if both a camera and a radar are used to detect obstacles, an attack on the visual component can be mitigated by the radar's independent detection capability, assuming the radar is not simultaneously compromised.

The concept of moving target defence (MTD), initially developed for cybersecurity applications, has been adapted for defending ML models in sensor fusion systems. By dynamically altering the configuration or parameters of the sensor fusion algorithm, it becomes harder for attackers to maintain the efficacy of their adversarial examples. Zhang suggested that varying the weights and the fusion method periodically or in response to suspected adversarial activities could disrupt targeted attack strategies [67].

3.2. Attack Taxonomy

3.2.1. Taxonomy of Adversarial Attacks

Adversarial attacks in AI systems can be classified based on multiple axes [76]:

- **Lifecycle Phase Affected**: Attacks can target the AI lifecycle phases of development, training, or deployment.
- **Compromised AI Trustworthiness Attributes**: These include security aspects such as accuracy, fairness, privacy, reliability, and robustness.
- **Impact Level**: Evaluating how much control or influence the attacker gains by exploiting the vulnerability, ranked from minimal (AI functions as intended) to total control (AI performs unintended, out-of-bounds actions).

3.2.2. Taxonomy Based on Attacker's Knowledge

Adversarial attacks can also be grouped by the attacker's knowledge [76]:

- **White-Box Attacks**: The attacker has complete access to the AI model, including architecture and training data. Attacks like FGSM and PGD fit here.
- **Black-Box Attacks**: The attacker has no internal knowledge and relies on querying the model to infer its behaviour. ZOO (Zeroth-Order Optimization) is an example.

- **Gray-Box Attacks**: The attacker has partial knowledge of the model, with some access to either data or internal parameters.

3.2.3. Compromised Attributes in Adversarial Attacks

Key trustworthiness attributes that adversarial attacks typically compromise are [76]:

- **Accuracy**: Attacks that reduce the model's ability to correctly map inputs to outputs.
- **Fairness**: Attacks that introduce bias in decision-making, leading to unequal treatment.
- **Privacy**: Attacks that reveal sensitive data or allow unauthorised access to model parameters or user data.
- **Reliability**: Attacks that degrade the model's consistency in producing expected outcomes.
- **Robustness**: Attacks that make the model susceptible to performance degradation or failure when exposed to adversarial inputs.

3.2.4. Classification of Adversarial Attacks

Below is a classification of common adversarial attacks based on the attacker's knowledge and the lifecycle phase impacted.

Attack	**Attacker Knowledge**	**Lifecycle Phase**	**Trustworthiness Attribute Compromised**
FGSM (Fast Gradient Sign Method)	White-box	Deployment	Accuracy, Robustness
PGD (Projected Gradient Descent)	White-box	Deployment	Accuracy, Robustness
CW (Carlini & Wagner)	White-box	Deployment	Privacy, Accuracy, Robustness
AdvGAN (Adversarial GAN)	Black-box, Gray-box	Training, Deployment	Privacy, Accuracy, Robustness
DeepFool	White-box	Deployment	Accuracy, Robustness
NAT (Noise-Added Training)	White-box, Black-box	Training	Accuracy, Robustness
Adversarial Patch	Black-box	Deployment	Accuracy, Robustness

BIM (Basic Iterative Method)	White-box	Deployment	Accuracy, Robustness

Table 1. Classification of Adversarial Attacks based on Attack Knowledge, Lifecycle Phase, and Compromised Attributes

3.3. Understanding Adversarial Attacks

3.3.1. Distance Metrics

Distance metrics are essential for measuring the discrepancy between the genuine sensor outputs and those manipulated by adversarial attacks. These metrics help in quantifying the extent of perturbations and their effectiveness in deceiving sensor fusion systems.

- **Euclidean Distance**: This metric, also known as the L2 norm, measures the straight-line distance between the original and adversarial sensor outputs. It is defined as:

$$\|\mathbf{z} - \mathbf{z}'\|_2 = \sqrt{\sum_{i=1}^{n} (z_i - z_i')^2} \tag{6}$$

  where $\mathbf{z}$ and $\mathbf{z}'$ represent the vector of original and perturbed sensor measurements, respectively.

- **Manhattan Distance**: Also known as the L1 norm, it measures the sum of the absolute differences between the two vectors. It is particularly useful for assessing perturbations in systems where changes are sparse but significant:

$$\|\mathbf{z} - \mathbf{z}'\|_1 = \sum_{i=1}^{n} |z_i - z_i'| \tag{7}$$

- **Cosine Similarity**: Unlike the other two, cosine similarity measures the cosine of the angle between two vectors, providing insight into the change in orientation rather than magnitude:

$$\cos(\theta) = \frac{\mathbf{z} \cdot \mathbf{z}'}{\|\mathbf{z}\| \|\mathbf{z}'\|} \tag{8}$$

3.3.2. Loss Functions

Loss functions quantify the error between the predicted and actual outputs of a model, guiding the learning process to mitigate the impact of adversarial perturbations.

- **Mean Squared Error (MSE)**: Predominantly used in regression tasks within sensor fusion models, it measures the average of the squares of the errors—that

is, the average squared difference between the estimated values and what is estimated:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\psi_i - \hat{\psi}_i)^2 \tag{9}$$

where ψ_i is the true fused output, and $\hat{\psi}_i$ is the predicted fused output by the sensor fusion algorithm.

- **Cross-Entropy Loss**: This loss function is essential for classification tasks in sensor fusion, especially when determining the likelihood of adversarial classes versus legitimate classes:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{10}$$

Here, $y_{i,c}$ is the binary indicator (0 or 1) if class label c is the correct classification for observation i, and $\hat{y}_{i,c}$ is the predicted probability output by the fusion system.

3.3.3. Gradient-Based Adversarial Attacks

One of the most well-known adversarial attacks is the Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al. [27]. The FGSM leverages the gradient of the loss function with respect to the input data to find the direction in which to perturb the input to maximise the loss [27]. The goal is to maximise the model's prediction error. This is formulated by adding a perturbation $\boldsymbol{\delta}$ to the input $\mathbf{z}$ such that the model's loss $L(\theta, \mathbf{z}, y)$ is maximised, where θ are the model (network) parameters, $\mathbf{z}$ is the input, and $\mathbf{y}$) is the true label or the target associated with $\mathbf{z}$.

Extending FGSM to Multi-Modal Data

Single Sensor Attack

In a multi-modal setup where data from multiple sensors or sources are used, a single sensor attack focuses on creating an adversarial example by perturbing the data from only one sensor, while keeping the data from other sensors unchanged. This approach can be effective if the model heavily relies on the attacked sensor for its decision making.

The adversarial example for this sensor can be formulated as:

$$z' = z + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y))$$

where:

- z are the sensor inputs: $[z_1, z_2, \ldots, z_n]$.
- x could be $\mathbf{z}$ or $\mathbf{h}_{\text{fused}}$ depending on the stage of fusion.
- ϵ is the perturbation magnitude.

- $\nabla_x L(\theta, x, y)$ is the gradient of the loss function L with respect to x, given model parameters θ and true label y.

Multiple Sensor Attack

For a multi-sensor attack, the objective is to coordinate perturbations across multiple sensors to create a more effective adversarial example. This might involve calculating the gradients for each sensor input and then optimising the perturbations to maximise the loss, potentially taking into account the relative influence of each sensor on the decision-making process.

Compute the gradients of the loss function with respect to the inputs of each sensor. This involves determining how sensitive the model's predictions are to changes in each sensor's input:

$$\nabla_{x_i} L(\theta, x, y) \text{ for each sensor } i$$

Decide how to balance or weight the perturbations across different sensors. This could be based on the relative importance of each sensor to the model's decision-making process or the vulnerability of each sensor to adversarial attacks. There are a few strategies that might be employed:

- **Uniform Perturbation:** Apply the same perturbation magnitude across all sensors.
- **Weighted Perturbation:** Apply different perturbation magnitudes based on the gradient magnitudes or the sensitivity of the model to each sensor.
- **Optimal Allocation:** Use optimisation techniques (e.g., linear programming, gradient-based optimisation) to determine the most effective perturbation distribution between sensors.

Now, apply the perturbations computed for each sensor to generate the adversarial examples. This step must be executed in a way that maximises the overall impact on the model:

$$z_i' = z_i + \epsilon_i \cdot \text{sign}(\nabla_{x_i} L(\theta, x, y)) \text{ for each sensor } i$$

where ϵ_i could be uniform or vary between sensors.

Often, a single pass through these steps may not yield the optimal adversarial example, especially in complex models with non-linear behaviours and dependencies between inputs. An iterative process might be necessary, where perturbations are incrementally adjusted based on feedback from the model's output or additional constraints (like evasion of detection mechanisms). Tools like gradient descent can be adapted to optimise the adversarial perturbations across multiple inputs simultaneously, ensuring that changes in one sensor are effectively synergizing with changes in others to lead to the highest possible increase in the model's loss.

Beyond FGSM, other gradient-based attacks, such as the Basic Iterative Method (BIM) and Projected Gradient Descent (PGD), apply iterative approaches to refine

the adversarial perturbation further. The BIM extends FGSM by applying it multiple times with a smaller step size α. This iterative process allows the perturbation to be more finely tuned to maximise the loss. PGD is a more robust version of BIM where after each update step, the perturbed input is projected back onto the l_∞-ball of radius ϵ around the original input. The l_∞-norm measures the maximum change in any single feature of the input, ensuring that none of the individual feature perturbations exceed ϵ. Alternatively, the l_2-norm could be used, which measures the overall Euclidean distance between the original and perturbed input, ensuring that the total change across all features stays within a certain bound.

3.3.4. Generative Adversarial Networks

Generative Adversarial Networks (GANs) have emerged as a powerful class of machine learning models that can generate realistic data from random noise through the use of adversarial training. Introduced by Goodfellow et al. [36], GANs consist of two components: a generator and a discriminator. The generator aims to produce data indistinguishable from real samples, while the discriminator attempts to differentiate between genuine data and the synthetic data generated by the generator. This adversarial dynamic drives the generator to produce increasingly convincing data, pushing both components to improve.

In the context of adversarial attacks, GANs can be employed to craft adversarial examples that deceive machine learning models. By exploiting the ability of GANs to approximate complex data distributions, they can generate perturbations that are subtle but highly effective at misleading models. The use of GANs in adversarial attacks presents a significant challenge for model robustness, especially in scenarios where attackers have limited knowledge of the model's internal workings, as seen in black-box attack settings.

Obtaining Information about the Model

In white-box attacks, the attacker's access to the model's parameters and architecture is assumed, which facilitates the generation of effective adversarial examples directly. In black-box attacks, the attacker typically relies on querying the model and observing the outputs to infer useful information. The attacker can submit a series of carefully crafted inputs and analyse the outputs to gain insights into the model's behaviour. This process may involve statistical analysis, optimisation techniques, or machine learning methods to approximate the model's decision boundaries.

An attacker may instead train a surrogate model that mimics the target model by using the input-output pairs obtained from queries. Adversarial examples generated against the surrogate model can often be transferred to the target model because of the similarity in decision boundaries.

Adversarial Generative Adversarial Network (AdvGAN)

AdvGAN, proposed by Xiao C. et al. [37], uses a generative adversarial network (GAN) to generate adversarial examples. The GAN is trained to produce examples

that maximise the loss of the target model without requiring explicit knowledge of the model's parameters or gradients.

GAN Architecture

- **Generator**: Generates adversarial examples from random noise or original inputs.
- **Discriminator**: Differentiates between real (original) inputs and fake (adversarial) inputs.
- **Target Model**: The pre-trained model to be attacked.

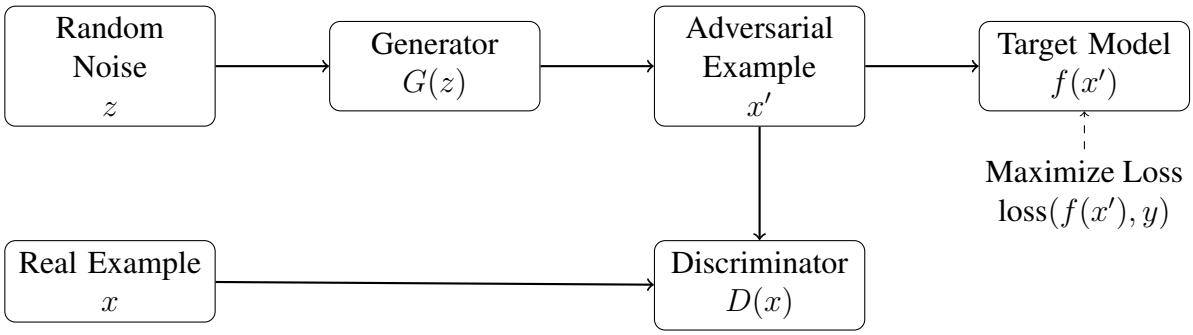


Figure 3. AdvGAN Architecture

The generator is trained to maximise the target model's loss for the adversarial examples.

$$\mathcal{L}_G = \mathbb{E}[\log(D(G(z)))] + \mathbb{E}[\text{loss}(f(G(x)), y)]$$

Here, D is the discriminator, G is the generator, z is the noise vector, x is the original input, f is the target model, and y is the true label. The discriminator is trained to correctly classify real inputs as real and adversarial inputs as fake.

$$\mathcal{L}_D = -\mathbb{E}[\log(D(x))] - \mathbb{E}[\log(1 - D(G(z)))]$$

Simultaneously, train the generator to produce adversarial examples and the discriminator to distinguish between real and adversarial examples. Use the trained generator to create adversarial inputs that deceive the target model.

3.4. Defence Against Adversarial Attacks in Machine Learning

Adversarial attacks in machine learning (ML) present significant challenges, especially in security-sensitive applications. Defending against these attacks requires a comprehensive understanding of the types of attacks and the development of robust defence strategies. Here, we focus on defence mechanisms against Fast Gradient Sign Method (FGSM) and other perturbation attacks, supported by mathematical justifications.

3.4.1. Input Preprocessing Techniques

Input preprocessing methods aim to mitigate adversarial perturbations by transforming the input data before it is processed by the model. These techniques can reduce the effectiveness of adversarial attacks by removing or diminishing the perturbations introduced into the sensor data. Later we demonstrate the ease of attacking extracted features vs the toughness of attacking a system that takes raw data as input. The Feature extraction pipeline itself normalises a lot of the data and tries to reduce the adversarial component of the vector space.

Feature Squeezing and Data transformation

Feature squeezing reduces the input data's complexity by limiting the available features for the adversary to exploit. This can be achieved by methods such as reducing the colour depth of images, applying spatial smoothing filters, or quantising sensor readings. In the context of sensor fusion systems, feature squeezing can be applied to each modality individually. For example, applying a median filter to depth images can smooth out small perturbations, while downsampling inertial data can reduce high-frequency noise introduced by adversarial attacks. By simplifying the input data, the model becomes less sensitive to small, adversarial changes.

Data transformations, such as JPEG compression, random resizing, or adding benign noise, can disrupt the structure of adversarial perturbations. These transformations can be particularly effective against attacks that rely on precise alterations to the input data. For multi-modal systems, modality-specific transformations can be applied. For instance, applying a Fourier transform to inertial data or random cropping to RGB images can help in mitigating adversarial effects.

3.4.2. Robust Model Architectures

Designing model architectures that are inherently more robust to adversarial perturbations is another approach to defence. This involves creating models that are less sensitive to small changes in the input data.

Use of Robust Activation Functions

Certain activation functions have been shown to enhance model robustness. For example, using activation functions like Swish or Mish [77] can improve the model's ability to generalise and resist adversarial perturbations compared to traditional ReLU functions. Integrating these activation functions into the neural networks for each modality may enhance their resilience.

Ensemble Methods

Ensemble methods involve training multiple models and combining their predictions to make a final decision. By aggregating outputs from diverse models, ensemble methods can mitigate the impact of adversarial attacks that might succeed against individual models but fail to deceive the ensemble as a whole [78]. In sensor fusion systems,

ensembles can be formed by training different models for each modality or by using various architectures for the same modality. Later we demonstrate how ensemble methods can make the system resistant to many attacks.

3.4.3. Cost of Defence

Implementing defence mechanisms against adversarial attacks in multi-modal sensor fusion systems involves various costs and trade-offs. These costs can manifest in terms of increased computational resources, extended training times, potential reductions in model performance on clean data, and added complexity in model implementation and maintenance. This section analyses the costs associated with three common defence strategies: adversarial training, gradient masking, and defensive distillation.

Adversarial Training

Adversarial training involves augmenting the training dataset with adversarial examples to improve the model's robustness against attacks [63]. The training process can be formalised as:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D} \left[\max_{\delta \in S} L(f_\theta(x+\delta), y) \right],$$

where:

- θ are the model parameters.
- x represents the input data.
- y are the true labels.
- D is the data distribution.
- δ is the adversarial perturbation within a permissible set S.
- L is the loss function.
- f_θ is the model function parameterized by θ.

The inner maximisation generates adversarial examples by finding perturbations δ that maximise the loss, while the outer minimisation updates the model parameters to minimise the loss on these adversarial examples.

Costs and Trade-offs:

- *Increased Computational Cost*: Adversarial training requires generating adversarial examples during each training iteration, which significantly increases computational demands. This is particularly taxing when using strong attacks like Projected Gradient Descent (PGD), which involve multiple gradient computations per iteration.
- *Extended Training Time*: The additional computations can lead to substantially longer training times. For large datasets or complex models, training can become impractically slow.

- *Resource Requirements*: Higher computational costs may necessitate more powerful hardware or cloud computing resources, increasing operational expenses.
- *Potential Reduction in Clean Data Accuracy*: Adversarial training may lead to a decrease in model performance on clean, non-adversarial data [79]. This trade-off between robustness and accuracy needs careful consideration.
- *Complexity in Implementation*: Integrating adversarial training into the existing training pipeline adds complexity, requiring careful tuning of hyperparameters like the strength of adversarial perturbations.

Gradient Masking

Gradient masking attempts to obscure the gradient information that attackers use to craft adversarial examples [80]. This can involve using non-differentiable layers, adding noise to gradients, or employing models with saturated activation functions.

Costs and Trade-offs:

- *Limited Effectiveness*: Gradient masking often provides a false sense of security. Attackers can circumvent it using gradient-free attacks or by approximating gradients numerically [81].
- *Model Performance Degradation*: Techniques used for gradient masking can negatively impact the model's ability to learn and generalise, potentially reducing accuracy on both clean and adversarial data.
- *Minimal Computational Overhead*: While gradient masking does not significantly increase computational costs, its limited effectiveness makes it a less desirable defence strategy.
- *Implementation Challenges*: Modifying the model architecture to mask gradients can introduce complexities and may affect the convergence of the training process.

Defensive Distillation

Defensive distillation trains a model to be less sensitive to small input perturbations by using softened outputs from a teacher model as labels [22]. The process involves:

1. Training a teacher model with a higher temperature T in the softmax function to produce soft labels:
$$\hat{y}^i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}},$$
where z_i are the logits, and T is the temperature parameter.
2. Training a student model using these soft labels, also with temperature T, to mimic the teacher model's output distribution.

Costs and Trade-offs:

- *Additional Training Phases*: Defensive distillation requires training both a teacher and a student model, effectively doubling the training workload.
- *Computational Overhead*: The increased training time and resource consumption can be significant, especially for large models and datasets.
- *Storage Requirements*: Maintaining both models increases memory usage and storage requirements.
- *Limited Robustness Improvement*: Defensive distillation has been shown to be ineffective against certain advanced attacks, such as the Carlini & Wagner (C&W) attack [71].
- *Implementation Complexity*: Incorporating distillation into the training pipeline adds complexity and requires careful tuning of the temperature parameter T.

Comparison and Overall Impact

Computational Costs:

- *Adversarial Training*: High computational cost due to adversarial example generation and longer training times.
- *Gradient Masking*: Minimal additional computational cost but limited effectiveness.
- *Defensive Distillation*: Moderate to high computational cost due to dual training phases.

Effectiveness Against Attacks:

- *Adversarial Training*: Generally effective against the types of attacks it is trained on but may not generalise to unforeseen attacks.
- *Gradient Masking*: Provides superficial robustness; vulnerable to gradient-free or black-box attacks.
- *Defensive Distillation*: Limited effectiveness against sophisticated attacks.

Impact on Model Performance:

- *Adversarial Training*: Potential decrease in accuracy on clean data; trade-off between robustness and performance.
- *Gradient Masking*: Possible degradation of overall model performance due to alterations in model architecture.
- *Defensive Distillation*: May maintain accuracy on clean data but offers limited robustness improvements.

Recommendations

Given the costs and trade-offs, the following recommendations can be made:

- **Adversarial Training** should be considered when robustness is a priority and resources allow for the increased computational demand. It is essential to balance the perturbation strength during training to minimise the impact on clean data accuracy.
- **Gradient Masking** is generally not recommended as a standalone defence due to its limited effectiveness and potential negative impact on model performance.
- **Defensive Distillation** may be used in conjunction with other defences but should not be relied upon solely for security against advanced attacks.

4. DESIGN & IMPLEMENTATION

This section details the design and implementation strategies employed for conducting adversarial attacks on machine learning (ML) sensor fusion systems. The core objective of this chapter is to explore the robustness of multi-modal sensor fusion models against adversarial perturbations. The experiments were conducted across various modalities, including Inertial, Skeleton, Depth, and RGB, attacking both raw sensor input and feature-extracted data.

Foolbox and the Adversarial Robustness Toolbox (ART) libraries were used to find adversarial examples, both comprehensive and open-source Python libraries designed for generating adversarial examples and evaluating ML models' resistance to attacks and training for a defence. The libraries support a wide range of perturbation techniques, making it ideal for this research. Multiple TensorFlow-based models, including Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs), are utilised to handle different data modalities and fusion strategies. Another key reason to employ these libraries was to leverage high-performance computing capabilities to train these models against the tried attacks.

The design approach integrates white-box (assuming model information is available) perturbation-based attacks (e.g. FGSM, PGD) and temporal delay attacks, specifically targeting the temporal aspect of sensor data. The study also tests a blackbox attack using HopSkipJump [82] for all modalities with the raw data classified using decision-level (late) fusion strategy. These attacks are applied at different levels of fusion, early, mid, and late, enabling a thorough analysis of the system's vulnerability across various integration points. The models are designed to process both raw sensor data and feature-extracted data, allowing us to investigate the efficacy of adversarial defences at multiple stages of the ML pipeline.

This chapter also outlines the implementation specifics, detailing how adversarial examples are crafted using Foolbox and ART, the configurations of TensorFlow models, and the deployment of defence mechanisms. The aim is to provide a comprehensive understanding of how adversarial attacks can compromise sensor fusion systems and how well different defences can mitigate these vulnerabilities.

4.1. Dataset

The dataset chosen for this study is the UTD-MHAD (University of Texas at Dallas - Multimodal Human Action Dataset) [83], which is publicly available and widely used in research. This dataset contains data from four different sensors: RGB cameras, depth sensors, inertial sensors, and a skeletal tracker. Its comprehensive coverage of different sensor modalities and its ability to capture a wide variety of human actions make it particularly suitable for studying sensor fusion systems in the context of adversarial attacks. With approximately ~861 data points across all modalities, it provides a rich foundation for exploring how multiple sensor inputs can be fused to enhance system performance, as well as how adversarial attacks can exploit the vulnerabilities of such systems.

A major factor in choosing the UTD-MHAD dataset is its extensive use in human action recognition and multi-modal sensor fusion research [84]. This dataset has

been widely adopted in prior studies, making it a well-researched topic with a wealth of reference models and techniques available for comparison. The availability of various models, ranging from traditional machine learning approaches to more advanced deep learning architectures like LSTM and CNN-based models, simplifies the implementation of adversarial attack experiments. With established baselines already in place, research can focus on experimenting with different adversarial attack strategies, such as perturbation-based and temporal delay attacks, without the need for extensive data preprocessing or model building from scratch. There is definitely some limitation on training Models on only one type of dataset as it leads to overfitting on the training dataset, but the study still applies if the training is extended to multiple MHAD datasets.

The dataset comprises recordings of 27 different human actions performed by 8 subjects, see Figure 4, leading to a diverse set of actions and variations. This diversity ensures the dataset captures a wide range of real-world variations in human activity, which is crucial for testing the general applicability of sensor fusion systems under adversarial conditions. Additionally, the multi-modal nature of the dataset allows for the study of how adversarial attacks affect each sensor type differently. For example, RGB and depth sensors may be more susceptible to visual perturbations, while inertial sensors might exhibit resilience but can be targeted through time-delay attacks. This wide sensor coverage is ideal for studying attack vectors and developing robust defence mechanisms across multiple modalities.

Furthermore, the UTD-MHAD dataset offers an excellent balance between accessibility and depth. While it is easy enough to conduct initial research on adversarial attacks due to the availability of existing models, it is also extensive enough to allow for a thorough investigation into the vulnerabilities and defence strategies of multi-modal sensor fusion systems. The richness of the dataset enables experimentation with various fusion techniques—whether early sensor-level fusion or late-stage feature and decision-level fusion—allowing for a comprehensive examination of how fusion strategies impact system resilience against adversarial attacks.

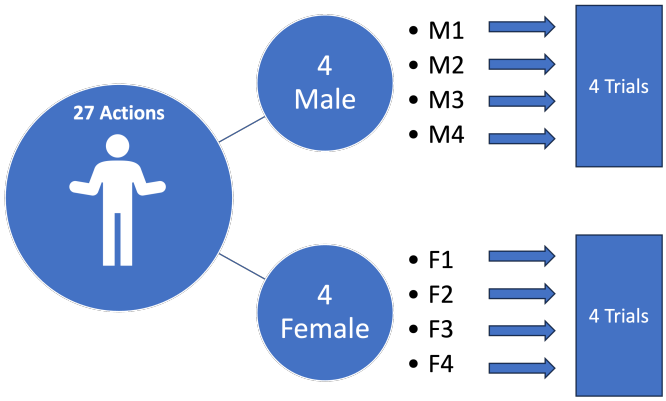


Figure 4. UTD-MHAD dataset

4.2. Implementation of Neural Network Models Using TensorFlow and Keras

This study implemented multiple models to evaluate the performance of multi-modal sensor fusion systems under adversarial attacks using the UTD-MHAD dataset. These LSTM and CNN models 2.6.1 handle data from different sensor modalities, employing independent architectures for each modality, along with various fusion strategies. Most sensor fusion models are implemented using these formats which make them a good set for this research. The following sections outline the key architectures used, ranging from single-modality models to fusion approaches, including early, hybrid, and late fusion. See 7.3 for details on the implementation.

4.2.1. Base Models, Preprocessing and Feature Extraction

Given the diversity of sensor data (RGB, depth, skeleton, and inertial), each modality requires specific processing techniques to normalise and structure the data appropriately. The preprocessing pipeline ensures that the data is in a format suitable for input into CNN, LSTM, and fusion models. Below is an overview of the key preprocessing steps and feature extraction methods implemented for each modality.

LSTM for Temporal Modelling (Raw RGB and Depth sensor data)

Long Short-Term Memory (LSTM) networks were employed to model the temporal dependencies in the raw RGB and Depth data. Some preprocessing is still done to reduce the input size, but no dimensionality reduction is performed, except conversion of 3-channel RGB video streams into 1-channel grayscaled versions. RGB streams provide detailed visual information, and LSTMs are ideal for capturing the long-range temporal dynamics of human movements.

Preprocessing steps involved, resizing and converting the frames to grayscale to reduce computational complexity while retaining the essential features for action recognition. Each RGB frame is first converted to grayscale using OpenCV (`cv2.cvtColor`) and then resized to 224x224 pixels. The processed grayscale frames are stored as 1-channel images and used as input for the LSTM model, which captures the temporal sequence in the data. Similarly, a region of interest is extracted (cropped) from the depth frames, focusing on the subject's movement area to reduce noise and improve the model's focus on relevant spatial information.

This allows the LSTM to capture time-dependent variations in human actions, such as continuous or subtle movements over time, providing a strong temporal model for recognising actions in video. Table 2 shows the performance of LSTM model on raw RGB and Depth data. This LSTM-based approach is critical for recognising time-varying patterns in human motion captured via RGB cameras. Same LSTM network was used to capture the temporal dynamics in depth sequences. Similar to the RGB model, this LSTM processes depth frames over time, allowing it to model the temporal evolution of movements based on depth maps. This approach is valuable when analysing how depth features change across consecutive frames, capturing time-dependent characteristics of human motion.

Table 2. Performance of LSTM for Temporal Modelling on RGB and Depth Data

Modality	Accuracy (%)	Precision	Recall	F1-Score
RGB (Grayscale)	61.2	0.71	0.61	0.65
Depth	71.3	0.82	0.79	0.80

CNN for Depth, Skeleton and Inertial Data

For the skeleton, depth and inertial sensor data, convolutional neural networks (CNNs) were employed to model the spatial patterns inherent in these modalities. Figure 5 shows the extracted 2-D images that are used to train these individual CNN models for Depth, Skeleton and Inertial modalities (left to right).

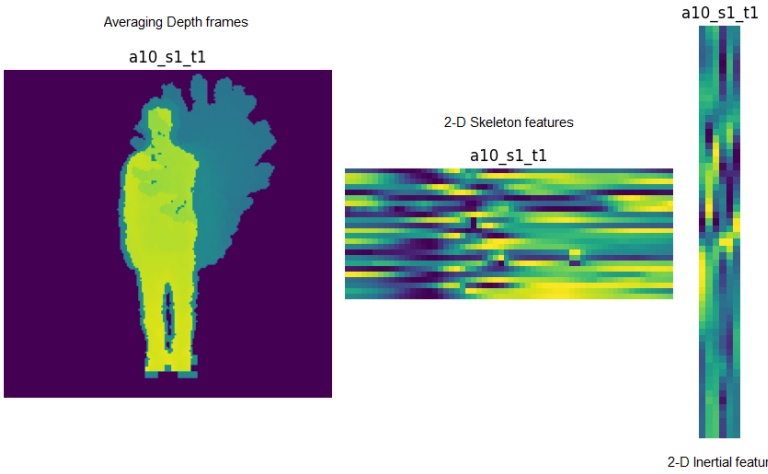


Figure 5. Feature extraction for Depth, Skeleton and Inertial data

The skeletal data, consisting of joint coordinates, were treated as structured spatial inputs, with CNNs learning the relationships between joints. The CNN identifies motion patterns and postures, which are crucial for understanding body movements during various actions. The skeleton data is normalised by subtracting the coordinates of a reference joint (e.g., the spine or torso) from all other joints. This transformation ensures that the skeleton data is independent of the subject's position in the frame and focuses on the relative movement of joints. These sequences are then interpolated to match a target frame count, ensuring consistency in the temporal data. This even sampling is especially important for modelling sequential data in CNNs.

Similarly, inertial data columns are interpolated to a fixed number of frames and normalised to ensure that all sensor readings are scaled consistently across time. This is essential for ensuring that the inertial data input is standardised for feature extraction models. Inertial sensor data was processed using the same CNN network. Although this data is time-series in nature, CNNs are used to extract structured patterns of acceleration and rotational movement, which are indicative of the subject's actions.

Depth frames are summarised by extracting the maximum and minimum depth values per pixel across frames, and an averaged depth map is generated. The resulting depth map is normalised to ensure that pixel values are scaled between 0 and 1. Depth features are extracted from the preprocessed depth maps, focusing on normalised spatial information.

Same CNN model is used to train for the early fusion case combining Depth, Inertial and Skeleton into 1 feature extracted 2-dimensional input. Table 3 shows performance of individual modalities trained on the same CNN architecture. CNN-based architectures are effective for capturing the spatial structure of both the skeleton and inertial data, focusing on movement patterns and postural changes.

Table 3. Performance of CNN for Depth, Skeleton, and Inertial Data

Modality	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
Skeleton	83.5	0.85	0.83	0.84
Depth	66.5	0.71	0.66	0.69
Inertial	85.8	0.87	0.85	0.86

4.2.2. Fusion Strategies

Early and Hybrid Fusion Models

In addition to independent models, early and hybrid fusion strategies were implemented to combine data from multiple sensors, enhancing the model's robustness by integrating complementary information from different modalities. 2.6.2. Results in table 4 clearly show that fusion has performed slightly better than the top per modality performance of the human-action recognition classification models.

Early fusion was implemented by combining skeleton, depth, and inertial data at the input level. The data from these modalities were concatenated into a unified input and processed using a shared CNN. This allows the model to learn joint representations from the start, capturing complementary information from all three sensor types and leveraging their combined strengths.

Hybrid/Intermediate Fusion (RGB with Early Fusion Model): The hybrid fusion model combines the features from the RGB modality (processed by the LSTM) with those extracted from the early fusion model (which combines skeleton, depth, and inertial data). After the independent models extract their respective features, these features are concatenated and passed through fully connected layers for the final classification. This hybrid fusion strategy takes advantage of both the detailed temporal features from the RGB data and the spatial-temporal information from the sensor fusion model, improving overall action recognition accuracy.

Table 4. Comparison of Independent and Fusion Models

Model	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
LSTM (RGB + Depth)	72.6	0.85	0.82	0.83
CNN (Skeleton + Inertial)	88.4	0.90	0.88	0.89
CNN Early Fusion	73.0	0.75	0.73	0.74
Hybrid Fusion	88.7	0.94	0.93	0.935

Late Fusion Using Raw and Extracted Feature Data (CNN)

A late fusion strategy was implemented, where each sensor modality is processed independently, and the fusion occurs at the decision level or feature level.

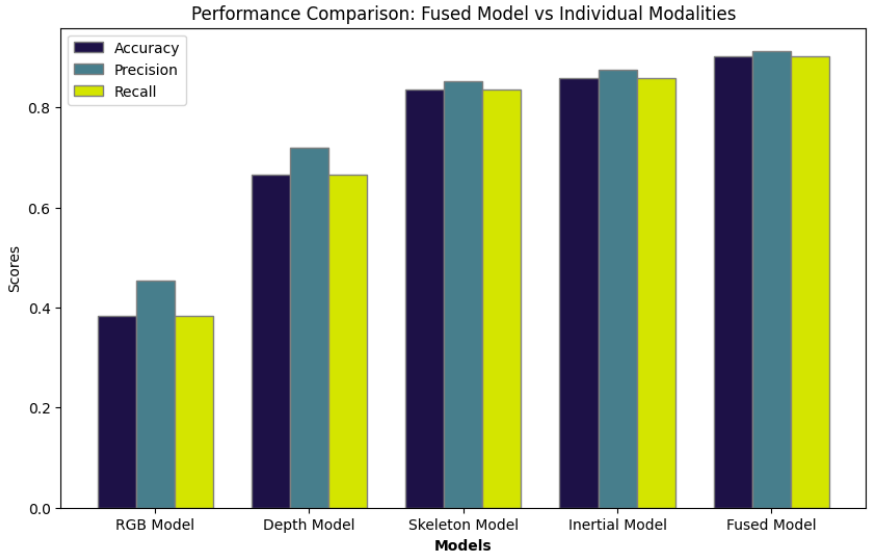


Figure 6. Bar chart showing performance of the late fusion strategy

Raw Feature Fusion (Late Fusion): In this late fusion model, the raw data from each sensor (RGB, depth, skeleton, and inertial) are processed independently by their respective models (LSTM for RGB, CNN for Depth, and CNN for Skeleton/Inertial). The individual models make predictions based on their respective data, and these predictions are fused at the decision level, using techniques like majority voting or weighted averaging. This late fusion ensures that each sensor contributes independently, anticipating that this will avoid any cross-modality vulnerabilities 2.3.4. As shown in Table 5 and Figure 6, decision level fusion (equal weighting) has performed better than any other methods of fusion. Though this might be due to the type of dataset and models chosen.

Extracted Feature Fusion (Late Fusion): Another late fusion model was implemented where high-level features extracted from each modality were concatenated instead of the final predictions. After feature extraction, the concatenated features were passed through fully connected layers to make the final classification. This fusion at the feature level allows for richer representations by combining complementary information from each modality before making a final prediction.

Table 5. Performance of Late Fusion Models (Decision and Feature Level)

Fusion Strategy	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
Decision-Level Fusion	90.5	0.91	0.90	0.91
Feature-Level Fusion	88.1	0.83	0.82	0.83

4.3. Implementation of Adversarial Attacks

This section details the methodologies and implementation strategies employed to conduct adversarial attacks on the multi-modal sensor fusion models developed in this study. The primary objective is to evaluate the robustness of these models against adversarial perturbations across various sensor modalities—Inertial, Skeleton, Depth, and RGB—and at different levels of data fusion. By leveraging state-of-the-art adversarial attack techniques and tools to expose potential vulnerabilities and assess the effectiveness of the fusion strategies under adversarial conditions. All adversarial attacks were implemented using Python, leveraging the Foolbox and Adversarial Robustness Toolbox (ART) libraries for standardised and efficient attack algorithms. The models were built using TensorFlow and Keras, allowing seamless integration with the adversarial attack frameworks. The Git repository for the AttackBench can be found at: https://github.com/ouspg/MLFusion-AttackBench.git. The experiments were conducted on high-performance computing clusters equipped with NVIDIA GPUs to handle the computational demands of training deep learning models and generating adversarial examples. This setup enabled to perform extensive evaluations across multiple attack scenarios and large datasets efficiently.

4.3.1. Overview of Attack Strategies

To comprehensively evaluate the models both white-box and black-box adversarial attacks are implemented, as well as temporal misalignment attacks specifically designed for time-series data. The attacks were applied to both raw sensor inputs and feature-extracted data, targeting the models at different fusion levels: early, intermediate (hybrid), and late fusion. Table 6 provides a summary of the attack strategies employed.

Table 6. Overview of Implemented Adversarial Attack Strategies

Attack Type	**Techniques Used**	**Targeted Modalities**	**Fusion Levels**
White-Box (Features)	FGSM	All Modalities	Early, Intermediate, Late
Black-Box (Raw)	HopSkipJump	Depth, Skeleton, Inertial	Late (Decision-Level)
Temporal Misalignment	Time-Delay Attacks	All modalities	Late

4.3.2. Evaluation Metrics

To quantify the effectiveness of the adversarial attacks and the robustness of the models the following evaluation metrics are employed:

- **Attack Success Rate (ASR)**: The percentage of adversarial examples that successfully cause misclassification.
- **Perturbation Magnitude**: Measured using norms such as L_2, indicating the size of the perturbations.

- **Model Accuracy Under Attack**: The classification accuracy of the models when subjected to adversarial examples.
- **Robustness Curve**: Plotting the model accuracy against varying perturbation magnitudes to visualise the degradation in performance.

These metrics were calculated for each attack type, modality, and fusion level. The results are presented in the next section, accompanied by graphs and charts illustrating the impact of the adversarial attacks on the models' performance.

4.3.3. White-Box Attacks Using Foolbox

White-box attacks assume full knowledge of the model architecture and parameters, allowing the attacker to craft perturbations that are highly effective in deceiving the model. The study utilises the Foolbox library [85], an extensive Python library for generating adversarial examples, to implement the following white-box attacks:

Fast Gradient Sign Method (FGSM)

The study applies FGSM 3.3.3 with multiple ϵ values to assess the model's sensitivity to varying degrees of perturbation. The attacks were conducted on all modalities individually and on fused models at different fusion levels. This gives an idea of a range of possible strength of perturbations that can be applied to the 2D feature images trying to apply the least amount of perturbation while still achieving high attack output. The accuracy vs relative epsilon is shown in Figure 7. Relative epsilon is normalised for all modalities:

$$\epsilon = (0,1) \cdot \text{relative\_epsilon}$$

Applying just $1 - 2\%$ change in pixel values can result in drastic drop in accuracy's for Individual modalities. During the fusion step these adversarial examples can be targeted for individual modalities or for all. Currently, average weighting has been applied at the fusion step. We examined how it performed when the attack was done on 1 modality with late fusion. Table 7 shows how the fusion model performs at different stages of fusion as discussed in 4.2.2. Attack level refers to the relative_epsilon %.

Table 7. **Fusion Model Accuracies under FGSM Attack**

Table 8. Fusion Model Accuracies (%) under FGSM Attack on Individual Modalities

Attack Level	**Early Fusion**		**Late Fusion**	
	Depth Attack	**Inertial Attack**	**Depth Attack**	**Inertial Attack**
0	86	86	84	84
1	82	81	84	82
2	79	77	82	81
3	76	71	82	77
4	74	67	81	74
5	73	61	80	72
6	72	56	80	68
7	72	53	79	66
8	72	47	78	63
9	71	42	77	60

Table 9. Estimated Fusion Model Accuracies (%) under FGSM Attack on All Modalities

Attack Level	**Early Fusion**	**Late Fusion**
0	86	84
1	81	82
2	77	78
3	72	70
4	68	68
5	61	63
6	58	58
7	54	50
8	49	48
9	41	46

Table 10. Comparison of Fusion Strategies Under FGSM Attack on Fused (Skeleton+Depth+Inertial) Modalities

Attack Level	**Early Fusion Accuracy (%)**	**Late Fusion Accuracy (%)**
0	86	78
1	81	78
2	76	78
3	71	75
4	67	73
5	62	70
6	57	66
7	53	64
8	50	61
9	48	59

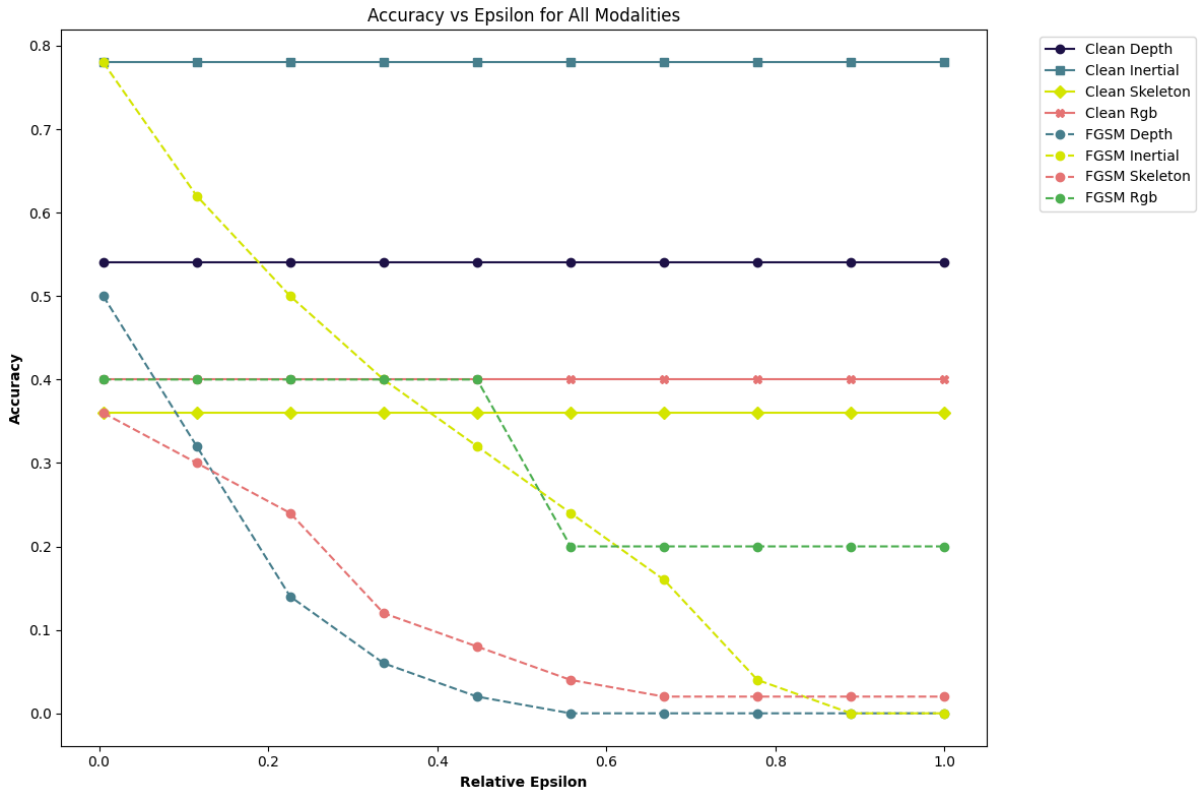


Figure 7. FGSM attack for individual modalities

4.3.4. Black-Box Attacks Using HopSkipJump

For scenarios where the model parameters are unknown, the study employs the HopSkipJump attack [82], a decision-based black-box attack that requires only the final class labels to craft adversarial examples. HopSkipJump is an optimisation-based attack that iteratively queries the model to approximate the decision boundary and generate perturbations. Figure 8 shows the performance of HopSkipJump on different modalities attacked at the raw input. It targets the raw data of Depth, Skeleton, and Inertial modalities, where individual modalities may be fused at the end. By attacking the raw data, we aim to assess the model's robustness when only the inputs and output labels are accessible to the attacker.

The HopSkipJump attack, implemented using ART, requires only the model's output labels to generate adversarial examples. It iteratively queries the model to approximate the decision boundary and create perturbations. Key parameters include whether the attack is targeted or untargeted, the norm (L2 or L^∞) used to measure perturbation size, and limits on iterations ("max_iter") and model queries ("max_eval"). Other important factors are the initial evaluation size, batch size for parallel processing, and verbosity for logging. This attack was used to evaluate the robustness of sensor fusion models by targeting raw data from Depth, Skeleton, and Inertial modalities.

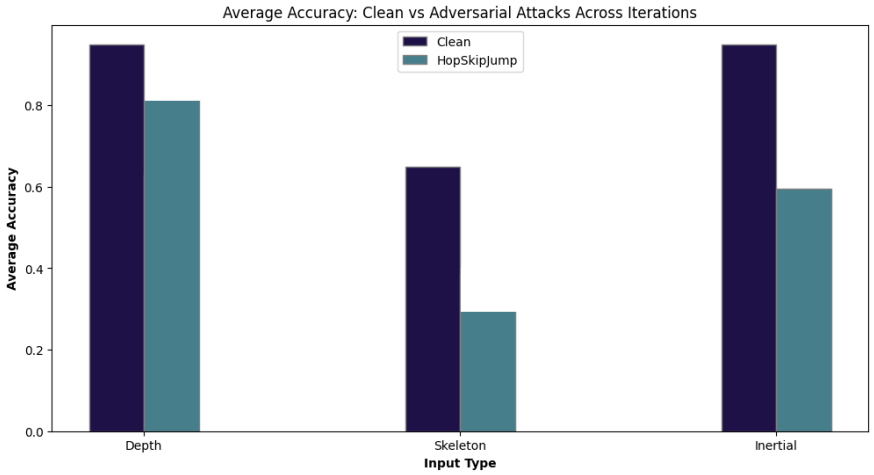


Figure 8. Black-box attack on individual modalities

4.3.5. Temporal Misalignment Attacks

Given that Inertial and Skeleton data are time-series in nature, the study designs temporal misalignment attacks that introduce time-delay perturbations to the sequential data. These attacks disrupt the temporal dependencies learned by the models, potentially leading to misclassification.

Time-Delay Perturbations

Time-delay attacks involve shifting the time-series data by a certain delay or altering the time intervals between data points. Specifically, the study implements:

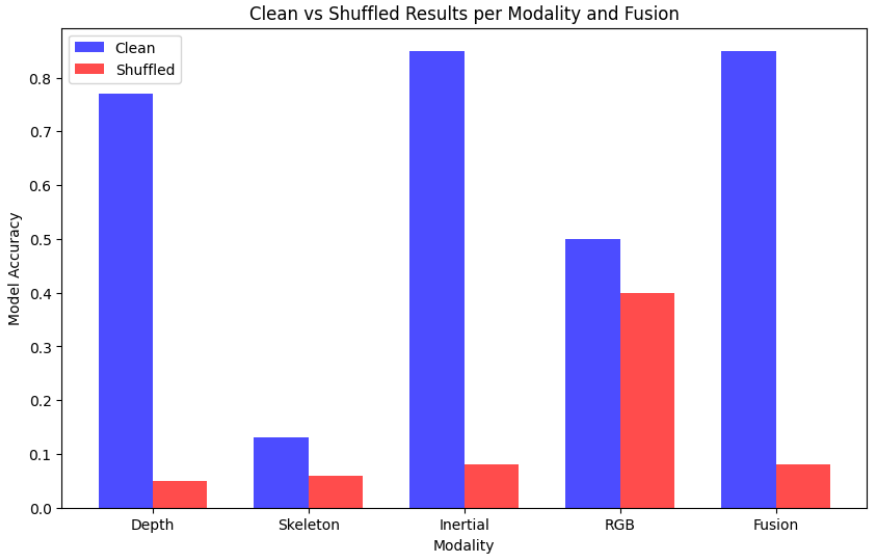


Figure 9. Random shuffle results

- **Random Shuffle**: Introducing random shuffles at different points in the sequence to disrupt the temporal consistency.

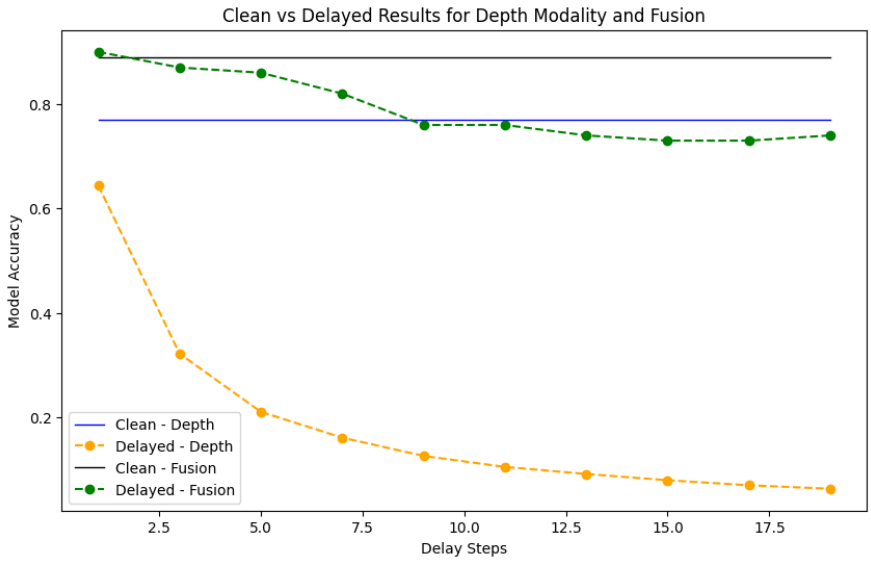


Figure 10. Only Delaying the Depth samples

- **Uniform Delay**: Shifting the entire sequence forward or backward by a fixed number of time steps, padding with zeros or truncating as necessary.

These perturbations were applied individually at raw fusion levels to evaluate the impact on the models' performance. The Fused output shown in Figure 9 shown is equally weighted across all modalities at the late fusion step. We can see the issue with equally weighted fusion. Even though the RGB output (LSTM) data is less damaged by the frame shuffles the final output is drastically low. Chart 10 shows if we only delay the depth sample but use smartly weighted predictions (Weighted Majority Voting) at the fusion step, the accuracy drop is not that worse as the model can deduce that there is noise or errors in some modalities input samples.

4.4. Defending Against Adversarial Attacks

This section outlines the defence mechanisms implemented to mitigate the impact of adversarial attacks on the multi-modal sensor fusion models. By leveraging adversarial training using the Adversarial Robustness Toolbox (ART) and implementing detection strategies with one-class Support Vector Machines (SVMs), the study aims to enhance the robustness and reliability of the models under adversarial conditions.

4.4.1. Adversarial Training with ART

Adversarial training is a widely adopted defence technique where the model is trained not only on the original dataset but also on adversarial examples [27]. This approach enables the model to learn representations that are robust to perturbations introduced by adversarial attacks.

Methodology

Using the ART toolbox [86], adversarial examples were generated from the feature extracted data during training and incorporated into the training data. Specifically, for each batch of training data, adversarial examples were crafted using the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) attacks. The models were then trained on a combination of original and adversarial samples, adjusting the loss function to penalise misclassifications on both types of data.

Implementation Details

The adversarial training was implemented for individual modalities as well as for fused models at different fusion levels. Key parameters for the adversarial example generation included:

- **Attack Methods**: FGSM with $\epsilon = 0{,}1$, PGD with $\epsilon = 0{,}05$, step size of 0.005, and upto 200 iterations.
- **Training Procedure**: For each epoch, adversarial examples were generated on-the-fly for the current batch and combined with the original data.
- **Loss Function**: A standard cross-entropy loss was used, with equal weighting for original and adversarial samples.

Results

Table 11 presents the performance of models after adversarial training. The models exhibited increased robustness, with a significant reduction in the attack success rate (ASR) under FGSM and PGD attacks compared to the models without adversarial training.

Table 11. Performance of Models After Adversarial Training

Model	Clean Accuracy (%)	Accuracy under Attack (%)	ASR Reduction (%)
LSTM (RGB)	61.2	28.5	24
CNN (Skeleton)	83.5	50.2	66
Early Fusion	73.0	29.1	34
Late Fusion	90.5	48.0	71

4.4.2. Adversarial Input Detection with One-Class SVMs

In addition to making the models robust through adversarial training, the study implements a detection mechanism to identify adversarial inputs before they reach the classifier. One-class SVMs are employed for this purpose, serving as an anomaly detection tool that distinguishes between normal and adversarial samples.

Methodology

One-class SVMs are trained exclusively on legitimate data to learn the boundary of normal input space. During inference, any input that falls outside this boundary is considered anomalous and potentially adversarial.

Implementation Details

For each modality, a one-class SVM was trained using the extracted features from the clean training data. The following parameters were used:

- **Kernel**: Radial Basis Function (RBF), enables the model to capture complex, non-linear relationships between features.
- **Nu**: 0.1 (an upper bound on the fraction of training errors).
- **Gamma**: Auto i.e. (1/num. of features)

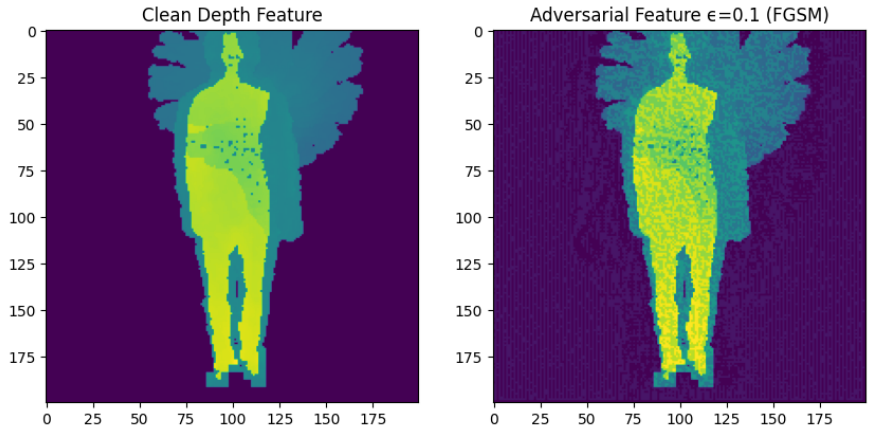


Figure 11. FGSM attack directly on Depth Feature

If we look at what the one-class svm trying to do is to detect perturbed features as shown in Figure 11.

During testing, the SVMs evaluated incoming samples and flagged those with decision scores below a certain threshold as adversarial. The thresholds were determined using a validation set to balance the true positive and false positive rates.

Results

Table 12. Adversarial Detection Performance using One-Class SVMs

Modality	TPR (%)	FPR (%)	Detection Accuracy (%)
Depth	85.0	10.5	87.2
Skeleton	88.5	8.0	90.0
Inertial	90.2	7.5	91.0
Early Fusion (above modalities)	83.0	12.0	85.5

The detection performance is evaluated using metrics such as True Positive Rate (TPR), False Positive Rate (FPR), and Detection Accuracy. Table 12 summarises the results.

5. SUMMARY

5.1. Discussion: Analysis of Adversarial Attacks and Defences

In this section, we analyse the outcomes of the design and implementation of various adversarial attacks and defence strategies in sensor fusion systems, as outlined in Chapter 4. The discussion focuses on the implications of different adversarial attacks, such as the Fast Gradient Sign Method (FGSM), black-box attacks (HopSkipJump), and temporal misalignment attacks, and the corresponding defences. By exploring these attack vectors, the strengths, and weaknesses of current defence strategies can be understood, which is crucial for improving system resilience.

Sensor fusion systems combine multiple sensor modalities to generate a holistic understanding of the environment. However, the integration of multi-modal data exposes these systems to vulnerabilities that attackers can exploit. The experiments demonstrated the ability of adversarial attacks to mislead the system by introducing perturbations that are either imperceptible or difficult to detect before fusion. These attacks can be as dangerous in multi-sensor systems configurations because adversarial perturbations may exploit sensor interactions.

5.1.1. Adversarial Attacks in Sensor Fusion Systems

The Fast Gradient Sign Method (FGSM) was used to perturb inputs from RGB cameras, depth sensors, and inertial measurement units (IMUs), revealing the vulnerability of the system to even minor perturbations. Specifically, in early fusion scenarios, where sensor data is concatenated prior to any processing, FGSM attacks caused significant misclassifications in the human action recognition tasks using the UTD-MHAD dataset. Early fusion, by its nature, amplifies the impact of adversarial inputs as it lacks intermediate validation checkpoints, allowing the attack to propagate unchecked through the system. This vulnerability was not as severe in intermediate fusion strategies, where some level of processing occurs before fusion.

The results highlight the importance of robust validation between sensor modalities. When all sensor data is fused without cross-checking, a single modality under attack can compromise the entire decision-making process. Therefore, the sensitivity of sensor fusion systems to perturbations suggests that stronger cross-validation and anomaly detection mechanisms are essential, especially for early fusion architectures. While FGSM exposed the brittleness of these systems, it is important to note that it represents a white-box attack. The attacker has access to the internal model parameters and the data processing pipeline, making the attack highly effective.

In contrast, black-box attacks like HopSkipJump, which do not require knowledge of the model's internal workings, presented a different set of challenges. The sensor fusion system showed resilience to minor perturbations but became vulnerable to larger distortions, particularly in depth sensors. HopSkipJump attacks were even effective in late fusion architectures, where decisions are made after each modality has been processed independently. Despite the system's robustness to white-box attacks, the success of black-box methods indicates that adversarial attacks can still pose a significant threat, even when model details are hidden from the attacker. The findings

suggest that late fusion, though more resilient in certain scenarios, is not impervious to black-box attacks, particularly when adversarial inputs are large enough to influence decision-making.

Temporal misalignment attacks introduced another level of complexity. By introducing delays in the depth sensor data relative to the RGB and IMU data, the experiments demonstrated how dependent sensor fusion systems are on precise temporal synchronisation. If the attacks are applied at the raw data level instead of the feature level, the damage is less. In applications such as human action recognition, where motion sequences are critical, even minor delays led to severe misclassifications. This underscores the need for time-aware defences that can identify and compensate for such misalignment's. Temporal attacks exposed a significant weakness in fusion systems, which rely heavily on synchronised multi-modal inputs.

5.1.2. Defense Strategies for Adversarial Attacks

To counter these adversarial attacks, several defence strategies were explored, each targeting specific vulnerabilities within the sensor fusion system. Adversarial training with multi-modal data, modality-specific adversarial detection, cross-modal consistency checking, and robust fusion strategies were implemented to assess their effectiveness.

Adversarial Training with Multi-modal Data aimed to improve system robustness by incorporating adversarial examples into the training dataset. This defence strategy showed a marked improvement in resistance to perturbation-based attacks such as FGSM and HopSkipJump, particularly in early fusion models. Adversarial training, though computationally expensive, significantly reduced misclassifications in the adversarial test sets, indicating that this defence can enhance resilience in the system. However, the training did not perform as well against temporal misalignment attacks. This suggests that adversarial training, while effective for perturbation-based attacks, may not be sufficient for addressing timing-related adversarial strategies, which highlights the need for complementary defences.

Modality-Specific Adversarial Detection introduced anomaly detectors trained for each sensor modality, which were integrated before the fusion stage to flag suspicious data. One-class Support Vector Machines (SVMs) were used to detect deviations from expected input patterns. This defence proved effective against black-box attacks, particularly by catching abnormal patterns in depth sensor data. While modality-specific detection provided a scalable and modular solution, its performance heavily relied on the accuracy of the anomaly detection models. The complexity of detecting subtle adversarial examples calls for more sophisticated feature extraction techniques, especially when dealing with black-box attacks, where perturbations might not be obvious at the input level.

Cross-Modal Consistency Checking, which compares data from different modalities to flag inconsistencies, emerged as an effective defence against temporal misalignment attacks. By comparing consistency scores between RGB, depth, and IMU inputs, the system was able to detect delays and misalignment's, reducing the effectiveness of temporal attacks. This low-complexity defence mechanism proved particularly valuable for systems where timing is crucial, as it required minimal computational

overhead and did not necessitate retraining models. However, it was less effective when adversarial attacks simultaneously affected multiple modalities.

Robust Fusion Strategies, such as weighted fusion and the use of robust statistical methods like the median instead of the mean, helped mitigate the influence of adversarially corrupted inputs. By assigning weights to sensor inputs based on their reliability, the system could downplay the impact of compromised modalities. Robust fusion strategies were most effective in scenarios where one sensor was compromised, but they struggled against attacks that targeted multiple sensors simultaneously. The results suggest that, while these strategies can improve system robustness, they are best employed alongside other defence mechanisms, such as cross-modal consistency checking or adversarial training. Late fusion architectures, when combined with ensemble methods, offered a higher degree of robustness compared to early and intermediate fusion approaches. Late fusion, where each modality is processed independently before decisions are combined, allows for more sophisticated decision-making strategies, such as weighted fusion or ensemble averaging. The experiments showed that ensemble methods, where multiple models are trained on different subsets of data or architectures, improved resilience against adversarial inputs. This defence mechanism ensured that no single model could dominate the decision-making process, reducing the overall impact of adversarial attacks.

A notable discovery from the experiments was the effectiveness of input transformation techniques in mitigating adversarial attacks. By applying transformations such as noise addition, resizing, and compression to input data, the system became more resilient to perturbation-based attacks. Techniques like Gaussian blurring and JPEG compression could disrupt the structure of adversarial inputs without degrading performance on clean data. These transformations proved especially useful in multi-modal sensor systems, where attacks often targeted a single modality. By applying transformations independently to each modality, the overall system was better protected against adversarial perturbations. The ease of integration and low computational complexity make input transformation a viable defence strategy, particularly when combined with more sophisticated methods like adversarial training. Another significant finding was the advantage of raw input fusion over feature-based fusion. Adversarial attacks often target extracted features, making systems reliant on early feature fusion vulnerable. By delaying fusion from raw input level to feature level, the system maintained the integrity of sensor data, making it harder for adversarial inputs to compromise the final output. This approach also allowed each modality to be processed independently, reducing the risk of adversarial perturbations spreading across modalities.

5.2. Conclusion

The research presented in this thesis has delved deeply into the vulnerabilities of machine learning-based sensor fusion systems to adversarial attacks and the corresponding defence strategies aimed at mitigating these risks. Sensor fusion systems, which integrate data from multiple modalities, such as RGB cameras, depth sensors, and inertial measurement units (IMUs), are increasingly applied in critical areas such as autonomous vehicles, surveillance, and human action recognition.

However, their complexity makes them highly susceptible to adversarial perturbations. This thesis investigated key adversarial attack methods, including the Fast Gradient Sign Method (FGSM), HopSkipJump black-box attacks, and temporal misalignment attacks, each of which revealed specific weaknesses within the fusion architectures.

The experimental results demonstrated that perturbation-based attacks like FGSM and C&W could significantly degrade the performance of sensor fusion systems, particularly when early fusion strategies are employed. These attacks exploit the system's reliance on individual sensor outputs, which, when fused without proper validation, can lead to catastrophic failures in decision-making. Black-box attacks such as HopSkipJump, which require no prior knowledge of the model's internal structure, were similarly successful in disrupting the system, highlighting the importance of developing robust defences that do not solely depend on the transparency of the model to the attacker.

Furthermore, the exploration of temporal misalignment attacks brought to light a unique set of vulnerabilities. In sensor fusion systems that rely on synchronised inputs from different modalities, even minor delays between the sensors can lead to significant degradation in performance. This was especially evident in human action recognition tasks, where precise temporal alignment between RGB and IMU data was critical for accurate classification. These findings underscore the importance of designing time-resilient architectures that can detect and compensate for temporal perturbations.

In response to these adversarial threats, several defence strategies were evaluated, each with its own strengths and limitations. Adversarial training, while effective in increasing system robustness against perturbation-based attacks, proved computationally expensive and less effective against temporal misalignment. Modality-specific adversarial detection, using anomaly detectors like one-class SVMs, provided a scalable solution to identifying attacks on individual sensor modalities, particularly against black-box attacks. However, this approach heavily relied on the quality of feature extraction, limiting its effectiveness in cases of subtle adversarial examples.

Cross-modal consistency checking emerged as an effective defence mechanism, particularly against temporal attacks. By comparing the consistency between inputs from different modalities, such as RGB and depth data, the system was able to flag misaligned or corrupted inputs before they could impact the fused output. This low-complexity defence, which required minimal changes to the overall system architecture, proved valuable in defending against a range of adversarial attacks, although it struggled against coordinated attacks that targeted multiple modalities simultaneously.

In addition to these defenses, the implementation of robust fusion strategies, such as weighted fusion and the use of robust statistics (e.g., the median), demonstrated that the impact of adversarial inputs could be mitigated at the fusion stage. These methods, however, were less effective in cases where all sensor inputs were compromised, indicating that more sophisticated fusion mechanisms are needed in such scenarios.

The research further explored the potential of input transformation techniques as a first-line defence. Techniques like Gaussian blurring and JPEG compression were recommended to disrupt adversarial perturbations effectively without significantly impacting the system's performance on clean data. The simplicity and low computational cost of these transformations make them a practical addition to

more comprehensive defence strategies, such as adversarial training or cross-modal consistency checking.

Late fusion architectures, particularly when combined with ensemble methods, emerged as the most robust approach to defending against a wide range of adversarial attacks. By processing each modality independently and combining the final decisions, late fusion allowed for more flexibility in handling corrupted inputs. Ensemble methods further enhanced robustness by ensuring that no single adversarial input could dominate the decision-making process.

In conclusion, while this thesis has explored several promising defense strategies, the evolving nature of adversarial attacks in sensor fusion systems requires ongoing research. The findings suggest that a multi-layered defense approach, combining adversarial training, input transformations, cross-modal consistency checks, and late fusion with ensemble methods, is necessary to build more resilient systems. As sensor fusion systems continue to play an increasingly vital role in critical applications, the development of more sophisticated defense strategies will be essential to ensuring their security and reliability in adversarial environments.

6. REFERENCES

[1] Hall D.L. & Llinas J. (1997) An Introduction to Multisensor Data Fusion. Proceedings of the IEEE .

[2] Mitchell H.B. (2007) Multi-sensor Data fusion: an Introduction. Springer.

[3] Varshney P.K. (2000), Multisensor Data Fusion.

[4] Thrun S., Burgard W. & Fox D. (2005) Probabilistic Robotics. MIT Press.

[5] Khaleghi B., Khamis A., Karray F.O. & Razavi S.N. (2013) Multisensor data fusion: A review of the state-of-the-art. Information Fusion 14, pp. 28–44.

[6] Klein L.A. (1999) Sensor and Data Fusion: A Tool for Information Assessment and Decision Making. SPIE Press Monograph Vol. PM138.

[7] Hassani S., Dackermann U., Mousavi M. & Liu H. (2024) A systematic review of data fusion techniques for optimized structural health monitoring. Information Fusion 103, p. 102136.

[8] Luo R.C. & Kay M.G. (1989) Multisensor integration and fusion in intelligent systems. IEEE Transactions on Systems, Man, and Cybernetics 19, pp. 901–931.

[9] Kibrete F., Woldemichael D.E. & Gebremedhen H.S. (2024) Multi-Sensor data fusion in intelligent fault diagnosis of rotating machines: A comprehensive review. Measurement 114658.

[10] Welch G. & Bishop G. (1995) An introduction to the Kalman filter. University of North Carolina at Chapel Hill.

[11] Urrea C. & Agramonte R. (2021) Kalman Filter: Historical Overview and Review of Its Use in Robotics 60 Years after Its Creation. Journal of Sensors 2021, pp. 1–21.

[12] Maybeck P.S. (1979) Stochastic models, estimation, and control (Vol. 1). Academic Press.

[13] Cilden-Guler D., Raitoharju M., Piche R. & Hajiyev C. (2019) Nanosatellite attitude estimation using Kalman-type filters with non-Gaussian noise. Aerospace Science and Technology 92, pp. 66–76.

[14] Yang C., Shi W. & Chen W. (2017) Comparison of Unscented and Extended Kalman Filters with Application in Vehicle Navigation. Journal of Navigation 70, pp. 411–431.

[15] Doucet A., De Freitas N. & Gordon N. (2001) Sequential Monte Carlo methods in practice. Springer Science \& Business Media.

[16] Li W., Wang Z., Yuan Y. & Guo L. (2016) Particle filtering with applications in networked systems: a survey. Complex \& Intelligent Systems 2, pp. 293–315.

[17] Ribeiro M. & Ribeiro I. (2004), Kalman and Extended Kalman Filters: Concept, Derivation and Properties.

[18] Levinson J., Askeland J., Becker J., Dolson J., Held D., Kammel S. & et al. (2011) Towards fully autonomous driving: Systems and algorithms. 2011 IEEE Intelligent Vehicles Symposium (IV) , pp. 163–168.

[19] Meng T., Jing X., Yan Z. & Pedrycz W. (2020) A survey on machine learning for data fusion. Information Fusion 57, pp. 115–129.

[20] Joseph A.D., Nelson B., Rubinstein B.I.P. & Tygar J.D. (2019) Adversarial Machine Learning. Cambridge University Press.

[21] Kurakin A., Goodfellow I. & Bengio S. (2017) Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 .

[22] Papernot N., McDaniel P., Wu X., Jha S. & Swami A. (2016) Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In: Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016, Institute of Electrical and Electronics Engineers Inc., pp. 582–597.

[23] Fogla P. & Lee W. (2006) Evading network anomaly detection systems: formal reasoning and practical techniques. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, Association for Computing Machinery, pp. 59–68.

[24] Wang Y., Sun T., Li S., Yuan X., Ni W., Hossain E. & Poor H.V. (2023), Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey.

[25] Dhesi S., Fontes L., Machado P., Ihianle I.K., Tash F.F. & Adama D.A. (2023), Mitigating Adversarial Attacks in Deepfake Detection: An Exploration of Perturbation and AI Techniques.

[26] Truong Phi, Duy Trung & Hoand Pham (2023) A Method Against Adversarial Attacks to Enhance the Robustness of Deep Learning Models. In: Integrated Uncertainty in Knowledge Modelling and Decision Making, Springer Nature Switzerland, pp. 346–357.

[27] Goodfellow I.J., Shlens J. & Szegedy C. (2015) Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 .

[28] Xu H., Ma Y., Liu H., Deb D., Liu H., Tang J. & Jain A.K. (2019) Adversarial Attacks and Defenses in Images, Graphs, and Text: A Review. International Journal of Automation and Computing 17, pp. 151–178.

[29] Balemans D., Casteels W., Vanneste S., De Hoog J., Mercelis S. & Hellinckx P. (2020) Resource efficient sensor fusion by knowledge-based network pruning. Internet of Things 11, p. 100231.

[30] Boulahia S.Y., Amamra A., Madi M.R. & Daikh S. (2021) Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. Machine Vision and Applications 32.

[31] Elmenreich W. (2007) Sensor fusion in time-triggered systems. Journal of Real-Time Systems 33, pp. 85–90.

[32] Jiang S., Shi J. & Moura S. (2024), A New Framework for Nonlinear Kalman Filters.

[33] Huang L., Joseph A.D., Nelson B., Rubinstein B.I.P. & Tygar J.D. (2011) Adversarial machine learning. Proceedings of the 4th ACM workshop on Security and artificial intelligence , pp. 43–58.

[34] Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D., Goodfellow I. & Fergus R. (2013) Intriguing Properties of Neural Networks. arXiv preprint arXiv:1312.6199 .

[35] Lal S., Rehman S.U., Shah J.H., Meraj T., Rauf H.T., Damaševičius R., Mohammed M.A. & Abdulkareem K.H. (2021) Adversarial Attack and Defence through Adversarial Training and Feature Fusion for Diabetic Retinopathy Recognition. Sensors 21.

[36] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A. & Bengio Y. (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680.

[37] Xiao C., Li B., Zhu J.Y., He W., Liu M. & Song D. (2018) Generating adversarial examples with adversarial networks. arXiv preprint arXiv:1805.07820 .

[38] Einicke G. (2012) Smoothing, filtering and prediction: Estimating the past, present and future. Optimal and Robust Estimation: With an Introduction to Stochastic Control Theory 3, pp. 1–10.

[39] Simon D. (2006) Optimal state estimation: Kalman, H infinity, and nonlinear approaches. John Wiley \& Sons.

[40] Groves P.D. (2013) Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems. IEEE.

[41] Lahat D., Adali T. & Jutten C. (2015) Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects. Proceedings of the IEEE 103, pp. 1449–1477.

[42] Samek W., Wiegand T. & Müller K.R. (2017) Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296 .

[43] Bhattacharyya S. (2023), Explaining Black Box Models: Ensemble and Deep Learning Using LIME and SHAP.

[44] Lipton Z.C. (2018) The Mythos of Model Interpretability. Communications of the ACM 61, pp. 36–43.

[45] Hendrycks D. & Dietterich T. (2019) Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1901.10513 .

[46] Castanedo F. (2013) A review of data fusion techniques. The Scientific World Journal .

[47] Demontis A. & others (2019) Hardening RGB-D object recognition systems against adversarial patch attacks. ACM Computing Surveys 52, pp. 1–33.

[48] Schmidhuber J. (2015) Deep learning in neural networks: An overview. Neural networks 61, pp. 85–117.

[49] LeCun Y., Bengio Y. & Hinton G. (2015) Deep learning. Nature 521, pp. 436–444.

[50] Goodfellow I., Bengio Y. & Courville A. (2016) Deep Learning Book. MIT Press.

[51] Gal Y. & Ghahramani Z. (2016) Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: Proceedings of The 33rd International Conference on Machine Learning, pp. 1050–1059.

[52] Hochreiter S. & Schmidhuber J. (1997) Long short-term memory. Neural computation 9, pp. 1735–1780.

[53] Lipton Z.C., Berkowitz J. & Elkan C. (2015) A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 .

[54] Papernot N., McDaniel P., Sinha A. & Wellman M. (2016) SoK: Towards the science of security and privacy in machine learning. In: 2016 IEEE European Symposium on Security and Privacy (EuroS\&P), pp. 399–414.

[55] Rosenberg I., Shabtai A., Elovici Y. & Rokach L. (2019) Defense Methods Against Adversarial Examples for Recurrent Neural Networks. CoRR abs/1901.09963.

[56] Cortes C. & Vapnik V. (1995) Support-vector networks. Machine learning 20, pp. 273–297.

[57] Xu H., Caramanis C. & Mannor S. (2009) Robustness and regularization of support vector machines. Journal of Machine Learning Research 10, pp. 1485–1510.

[58] You Z., Ye J., Li K., Xu Z. & Wang P. (2019) Adversarial Noise Layer: Regularize Neural Network by Adding Noise. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 909–913.

[59] Tu J., Li H., Yan X., Ren M., Chen Y., Liang M., Bitar E., Yumer E. & Urtasun R. (2021) Exploring Adversarial Robustness of Multi-Sensor Perception Systems in Self Driving. CoRR abs/2101.06784.

[60] Lin J., Dang L., Rahouti M. & Xiong K. (2021) ML Attack Models: Adversarial Attacks and Data Poisoning Attacks. CoRR abs/2112.02797.

[61] Papernot N., McDaniel P.D., Jha S., Fredrikson M., Celik Z.B. & Swami A. (2015) The Limitations of Deep Learning in Adversarial Settings. CoRR abs/1511.07528.

[62] Eykholt K., Evtimov I., Fernandes E., Li B., Rahmati A., Xiao C., Prakash A., Kohno T. & Song D. (2018) Robust Physical-World Attacks on Deep Learning Models. arXiv:1707.08945 [cs] .

[63] Madry A., Makelov A., Schmidt L., Tsipras D. & Vladu A. (2019) Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083 [cs, stat] .

[64] Athalye A., Engstrom L., Ilyas A. & Kwok K. (2018), Synthesizing Robust Adversarial Examples.

[65] Brown T.B., Mané D., Roy A., Abadi M. & Gilmer J. (2018) Adversarial Patch. arXiv:1712.09665 [cs] .

[66] Thys S., Ranst W.V. & Goedemé T. (2019) Fooling automated surveillance cameras: adversarial patches to attack person detection. CoRR abs/1904.08653.

[67] Zhang C., Yu S., Tian Z. & Yu J.J.Q. (2023) Generative Adversarial Networks: A Survey on Attack and Defense Perspective. ACM Computing Surveys .

[68] Gu T., Dolan-Gavitt B. & Garg S. (2019) BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv:1708.06733 [cs] .

[69] Bagdasaryan E., Veit A., Hua Y., Estrin D. & Shmatikov V. (2020) How To Backdoor Federated Learning. In: S. Chiappa & R. Calandra (eds.) Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, vol. 108, PMLR, vol. 108, pp. 2938–2948.

[70] Wu Z., Peng Y. & Wang W. (2022) Deep learning-based unmanned aerial vehicle detection in the low altitude clutter background. IET Signal Processing 16, pp. 588–600.

[71] Carlini N. & Wagner D. (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57.

[72] Hendrycks D. & Gimpel K. (2018), A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks.

[73] Meng D. & Chen H. (2017), MagNet: a Two-Pronged Defense against Adversarial Examples.

[74] Samangouei P., Kabkab M. & Chellappa R. (2018), Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models.

[75] Papernot N., McDaniel P. & Goodfellow I. (2016), Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples.

[76] Pispa A. & Halunen K. (2024) Comprehensive Artificial Intelligence Vulnerability Taxonomy. European Conference on Cyber Warfare and Security 23, pp. 379–387.

[77] Hendrycks D. & Gimpel K. (2016) Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. CoRR abs/1606.08415.

[78] Pang T., Xu K., Du C., Chen N. & Zhu J. (2019) Improving Adversarial Robustness via Promoting Ensemble Diversity. CoRR abs/1901.08846.

[79] Tsipras D., Santurkar S., Engstrom L., Turner A. & Madry A. (2018) Robustness May Be at Odds with Accuracy. arXiv: Machine Learning .

[80] Papernot N., Mcdaniel P., Goodfellow I.J., Jha S., Celik Z.B. & Swami A. (2016) Practical Black-Box Attacks against Machine Learning. Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security .

[81] Athalye A., Carlini N. & Wagner D.A. (2018), Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples.

[82] Chen J. & Jordan M.I. (2019) Boundary Attack++: Query-Efficient Decision-Based Adversarial Attack. CoRR abs/1904.02144.

[83] Chen C., Jafari R. & Kehtarnavaz N. (2015) UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In: 2015 IEEE International Conference on Image Processing (ICIP), pp. 168–172.

[84] Shaikh M.B., Chai D., Islam S.M.S. & Akhtar N. (2024) From CNNs to Transformers in Multimodal Human Action Recognition: A Survey. ACM Transactions on Multimedia Computing, Communications and Applications .

[85] Rauber J., Brendel W. & Bethge M. (2017) Foolbox: A Python toolbox to benchmark the robustness of machine learning models. CoRR .

[86] Nicolae M.I., Sinn M., Minh T.N., Rawat A., Wistuba M., Zantedeschi V., Molloy I.M. & Edwards B. (2018) Adversarial Robustness Toolbox v0.2.2. CoRR abs/1807.01069.

7. APPENDICES

7.1. Derivation of Posterior Distribution

Conditional probability is the probability of an event occurring given that another event has already occurred. We are interested in the conditional probability of a state $\mathbf{x}$ given a set of measurements $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$. The posterior probability of the state $\mathbf{x}$ given the measurements is derived using Bayes' theorem:

$$p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n) = \frac{p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \mid \mathbf{x}) p(\mathbf{x})}{p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)}$$

- $p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$ is the **posterior distribution**.
- $p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \mid \mathbf{x})$ is the **likelihood**, representing the probability of the measurements given the state.
- $p(\mathbf{x})$ is the **prior distribution** of the state, representing our knowledge about $\mathbf{x}$ before considering the measurements.

We derive the posterior distribution of the state $\mathbf{x}$ given the measurements $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$. Using Bayesian estimation, the posterior distribution of the state $\mathbf{x}$ given the measurements $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$ is:

$$p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n) \propto p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \mid \mathbf{x}) p(\mathbf{x})$$

Assuming independence between the measurements given the state, this can be decomposed as:

$$p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \mid \mathbf{x}) = \prod_{i=1}^{n} p(\mathbf{z}_i \mid \mathbf{x})$$

Therefore, the posterior distribution becomes:

$$p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n) \propto \left(\prod_{i=1}^{n} p(\mathbf{z}_i \mid \mathbf{x}) \right) p(\mathbf{x})$$

For a Gaussian noise model, $p(\mathbf{z}_i \mid \mathbf{x})$ is given by:

$$p(\mathbf{z}_i \mid \mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m |\mathbf{R}_i|}} \exp\left(-\frac{1}{2} (\mathbf{z}_i - \mathbf{h}_i(\mathbf{x}))^T \mathbf{R}_i^{-1} (\mathbf{z}_i - \mathbf{h}_i(\mathbf{x})) \right)$$

The maximum a posteriori (MAP) estimate $\hat{\mathbf{x}}$ is obtained by maximising the posterior distribution:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$$

Keywords: Bayesian Estimation, Gaussian Noise Model, Maximum a Posteriori (MAP) Estimation

7.2. Fused Estimation Error Covariance for Two Sensors

Consider two sensors with measurement functions $\mathbf{h}_1(\mathbf{x})$ and $\mathbf{h}_2(\mathbf{x})$ and noise covariances $\mathbf{R}_1$ and $\mathbf{R}_2$. The individual estimation error covariances are:

$$\mathbf{P}_1 = \mathbf{H}_1 \mathbf{R}_1 \mathbf{H}_1^T$$

$$\mathbf{P}_2 = \mathbf{H}_2 \mathbf{R}_2 \mathbf{H}_2^T$$

The fused estimation error covariance is:

$$\mathbf{P} = \left(\mathbf{H}_1^T \mathbf{R}_1^{-1} \mathbf{H}_1 + \mathbf{H}_2^T \mathbf{R}_2^{-1} \mathbf{H}_2 \right)^{-1}$$

To show that $\mathbf{P} \leq \mathbf{P}_1$ and $\mathbf{P} \leq \mathbf{P}_2$, we use the fact that the inverse of a sum of matrices is less than or equal to the sum of the inverses of the individual matrices:

$$(\mathbf{A} + \mathbf{B})^{-1} \leq \mathbf{A}^{-1} + \mathbf{B}^{-1}$$

Applying this to our error covariance matrices:

$$\mathbf{P} \leq \left(\mathbf{H}_1^T \mathbf{R}_1^{-1} \mathbf{H}_1 \right)^{-1} = \mathbf{P}_1$$

$$\mathbf{P} \leq \left(\mathbf{H}_2^T \mathbf{R}_2^{-1} \mathbf{H}_2 \right)^{-1} = \mathbf{P}_2$$

Therefore, the fused estimation error covariance $\mathbf{P}$ is less than or equal to the individual error covariances $\mathbf{P}_1$ and $\mathbf{P}_2$.

Keywords: Fused State Estimate, Measurement Noise, Uncertainty Reduction

7.3. Machine Learning Models, CNN and LSTM

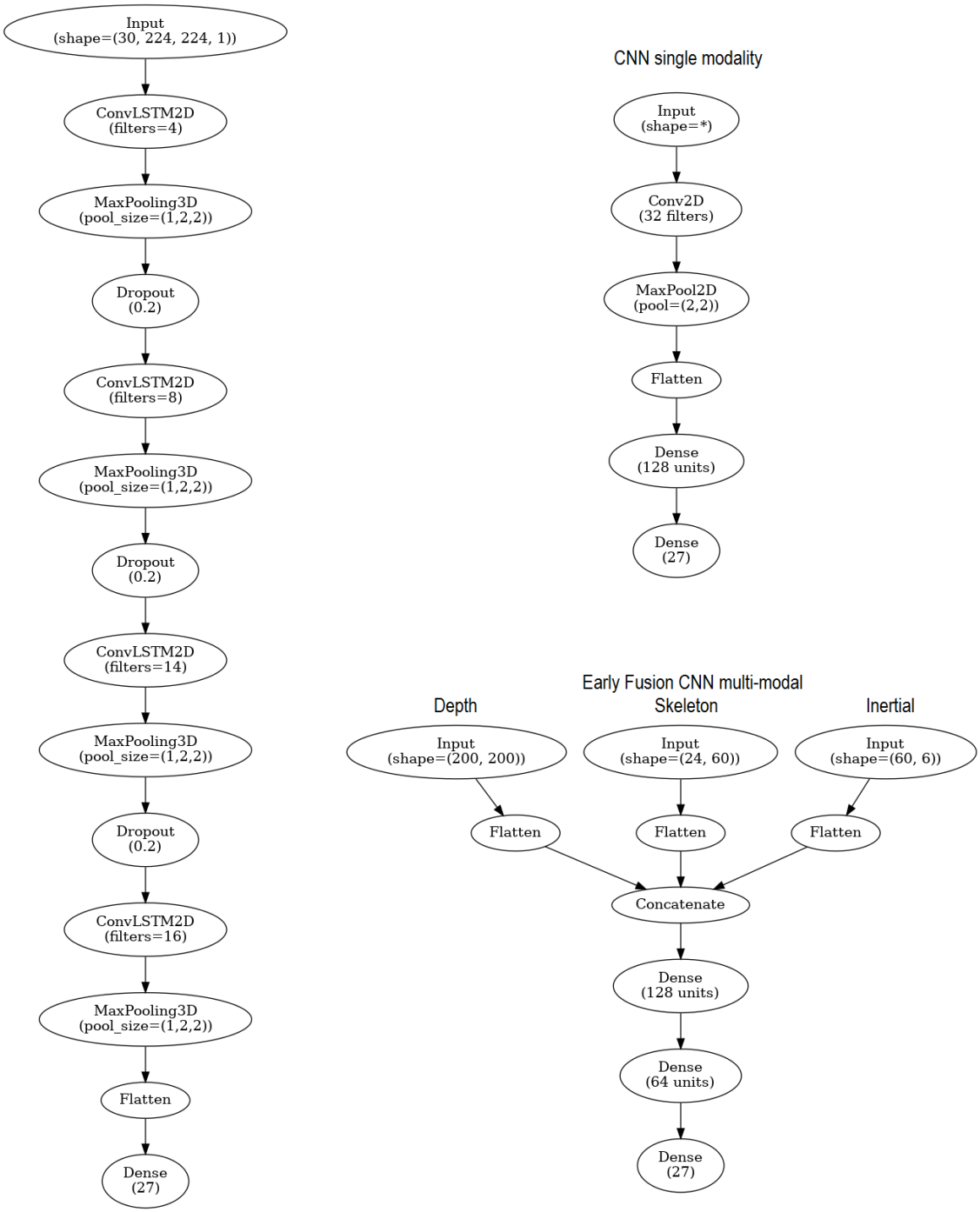


Figure 12. LSTM Model for RGB data, CNN Models for single and multi-modal inputs