FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Safe-DL: A Modular Framework for Evaluating and Mitigating Adversarial Threats in Deep Learning

Tiago Filipe Magalhães Barbosa

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Luís Paulo Reis

Second Supervisor: Prof. Luís Filipe Antunes

July 13, 2025

Safe-DL: A Modular Framework for Evaluating and Mitigating Adversarial Threats in Deep Learning

Tiago Filipe Magalhães Barbosa

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Daniel Castro Silva
Referee: Prof. Pedro Manuel Abreu

July 13, 2025

Resumo

A crescente adoção de sistemas baseados em redes neuronais profundas em domínios críticos, nomeadamente na medicina, mobilidade e segurança, levanta sérias preocupações quanto à sua robustez face a ataques adversariais. Apesar do seu elevado desempenho, estes sistemas revelam vulnerabilidades que podem ser exploradas de forma a comprometer decisões automatizadas. Embora existam soluções parciais para mitigar tais ameaças, o ecossistema atual caracteriza-se por abordagens fragmentadas, complexas e de difícil integração prática, muitas vezes desconectadas de orientações regulatórias emergentes como o *AI Act* da União Europeia.

Esta dissertação propõe a **Safe-DL**, uma framework modular, extensível e auditável para avaliação e mitigação de ameaças adversariais em sistemas de aprendizagem profunda. A solução organiza-se em seis módulos: modelação de ameaças, simulação de ataques, análise de risco, aplicação de defesas, avaliação da mitigação e geração de relatórios. Em conjunto, estes módulos permitem uma análise sistemática e integrada da segurança dos modelos.

A validação experimental foi realizada num cenário realista de classificação de imagens, com ataques de envenenamento de dados, backdoors e evasão. Os resultados demonstram a capacidade da framework para identificar vulnerabilidades e aplicar contramedidas eficazes, com resultados rastreáveis e de fácil interpretação.

Combinando reprodutibilidade, usabilidade e alinhamento com exigências ético-legais, esta framework representa um contributo prático e científico para o desenvolvimento de sistemas de inteligência artificial mais robustos, transparentes e confiáveis. O trabalho desenvolvido com a `Safe-DL` constitui uma base sólida para futuras extensões da framework a outras modalidades e contextos regulatórios.

Palavras-chave: Aprendizagem Profunda, Ataques Adversariais, Segurança em Redes Neuronais, Mecanismos de Defesa, Framework Modular, Safe-DL

Abstract

The increasing adoption of deep neural network-based systems in critical domains, such as medicine, mobility, and security, raises serious concerns regarding their robustness against adversarial threats. Despite their strong performance, these systems exhibit exploitable vulnerabilities that may compromise automated decision-making processes. While partial solutions to mitigate such threats exist, the current ecosystem is characterised by fragmented, complex, and hard-to-integrate approaches, often disconnected from emerging regulatory guidelines such as the European Union's *AI Act*.

This dissertation introduces **Safe-DL**, a modular, extensible, and auditable framework designed to evaluate and mitigate adversarial threats in deep learning systems. The solution is organised into six modules: threat modelling, attack simulation, risk analysis, defence application, mitigation assessment, and final reporting. Together, these modules enable a systematic and integrated security analysis of machine learning models.

Experimental validation was performed in a realistic image classification scenario, including data poisoning, backdoor, and evasion attacks. The results demonstrate the framework's ability to identify vulnerabilities and apply effective countermeasures, producing traceable and human-readable outputs.

By combining reproducibility, usability, and alignment with ethical and legal requirements, this framework offers a practical and scientific contribution toward building more robust, transparent, and trustworthy AI systems. The work developed using `Safe-DL` lays a solid foundation for future framework extensions to other modalities and regulatory contexts.

Keywords: Deep Learning, Adversarial Attacks, Neural Network Security, Defence Mechanisms, Modular Framework, Safe-DL

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) offer a global framework to address the world's most pressing challenges, from social equity and environmental protection to technological responsibility. As future engineers and researchers, reflecting on how our work can contribute to a more inclusive, ethical, and sustainable world is essential.

This dissertation aligns with this vision by contributing to several SDGs through the development of **Safe-DL**, a framework for systematically assessing and mitigating adversarial threats in deep learning systems. Beyond its technical contributions, the project promotes transparency, trustworthiness, and public access to critical Artificial Intelligence (AI) knowledge, all essential for building sustainable digital infrastructures and informed institutions.

The analysis below identifies the SDGs most directly impacted by this work, outlines the specific targets addressed, and presents measurable contributions and associated indicators.

The specific Sustainable Development Goals addressed in this dissertation are:

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 16 Promote peaceful and inclusive societies for sustainable development, provide access to justice for all and build effective, accountable and inclusive institutions at all levels

SDG	Target	Contribution	Performance Indicators and Metrics
4	4.3	`Safe-DL` is distributed as a free and open-source educational resource, accessible to students in higher education, promoting equitable access to practical AI security tools.	Public availability of the repository; Use in MSc projects or advanced courses
	4.4	The framework helps students and practitioners develop technical skills in AI threat modelling, attack simulation, defence strategies, and risk assessment.	Number of documented modules; Quality of usage examples; Adoption in academic or training contexts
	4.7	By integrating ethical and legal perspectives (EU *AI Act*), `Safe-DL` fosters awareness of responsible AI practices and sustainable digital development.	References to regulation in documentation/tutorials; Inclusion of ethical impact discussions in reports or deliverables
9	9.5	`Safe-DL` fosters applied research and technological innovation by providing a structured and extensible platform for adversarial robustness evaluation in deep learning systems.	Number of modules implemented; Use in academic research projects; Scientific output (e.g., dissertation, papers)
	9.b	As a modular and open-source tool, the framework supports local capacity-building in AI trust and safety, contributing to national innovation and development strategies.	Public code repository; Institutional or academic collaborations; Reuse in national Research and Development (R&D) initiatives
	9.c	By offering open access to a digital security framework, `Safe-DL` contributes to broader access to responsible AI technologies, particularly for educational and research use.	Accessibility of the tool; Clarity of documentation; Usage by non-specialist or under-resourced communities
16	16.6	`Safe-DL` promotes transparency and accountability in AI systems through traceable outputs, risk assessment reports, and auditable evaluation pipelines, supporting trustworthy digital institutions.	Generation of structured reports; Ability to explain and audit security decisions; Documentation enabling third-party review
	16.10	By offering open, accessible, and well-documented tools for AI safety, `Safe-DL` ensures public access to critical information on adversarial threats and mitigation techniques.	Open-source publication; Public tutorials and examples; Accessibility and clarity of outputs for diverse stakeholders

Acknowledgements

As I reach the end of this important chapter in my life, I feel deeply grateful to all those who, in one way or another, contributed to making this work possible. This acknowledgement is dedicated to all of you.

My most sincere thank you to my parents, **Carla** and **Jacinto**. You always gave me everything I needed: support, motivation, stability, and encouragement. Your dedication has been unwavering, and this journey would not have been possible without your constant presence. Mom, thank you for always being there with your tireless care and attention to detail. Dad, thank you for being an inspiration in choosing the field of computer science and for your genuine interest in everything I do. This achievement belongs to you as much as it does to me.

To my grandparents, **Laurinda** and **Arlindo**, I want to express a very special thank you. You have always been essential to my growth with your affection, wisdom, and unconditional support. It makes me truly happy to share this milestone with you.

To my girlfriend, **Mafalda**, thank you for standing by me throughout this process. Your support, patience, and company helped me maintain balance and focus. And, of course, our tennis and padel matches were perfect moments to relax and recharge during the more intense phases of this project. Thank you for being part of this journey.

To my friends and extended family, thank you for your encouragement, presence, and the light moments that reminded me there's more to life than thesis writing. I dedicate this accomplishment to you as well.

To my advisors, **Professor Luís Paulo Reis** and **Professor Luís Filipe Antunes**, I am grateful for your guidance, availability, and invaluable contributions throughout this dissertation. I also extend a special thanks to **LIACC** and Professor Luís Paulo Reis for providing access to a high-performance GPU machine, which significantly facilitated the development and testing of the framework.

To everyone who, directly or indirectly, played a role in this journey, my heartfelt thanks.

Tiago Barbosa

"Would I rather be feared or loved? Easy. Both.
I want people to be afraid of how much they love me."

Michael Scott

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
ART	Adversarial Robustness Toolbox
ASR	Attack Success Rate
CAD	Clean Accuracy Drop
CLI	Command-line Interface
CNN	Convolutional Neural Network
C&W	Carlini & Wagner
DCS	Defence Cost Score
DL	Deep Learning
DNN	Deep Neural Network
DP	Differential privacy
FGSM	Fast Gradient Sign Method
FLOPs	FLoating-point Operations Per Second
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
MS	Mitigation Score
MLaaS	Machine Learning as a Service
MLOps	Machine Learning Operations
NES	Natural Evolution Strategies
PCB	Printed Circuit Board
PGD	Projected Gradient Descent
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
SPSA	Simultaneous Perturbation Stochastic Approximation

Chapter 1

Introduction

Artificial Intelligence (AI), particularly deep neural networks (DNNs), has enabled significant advances in fields such as medical imaging, finance, and autonomous driving. However, these models remain alarmingly vulnerable to adversarial perturbations that can cause critical failures in safety-sensitive scenarios. To address this issue, we introduce **Safe-DL**. This modular framework supports practitioners with or without deep learning (DL) expertise through the whole security workflow, including threat modelling, attack simulation, risk scoring, automated defence selection, and structured reporting, all in a reproducible and transparent manner. Grounded in principles of accountability and explainability promoted by upcoming regulations such as the European Union's *AI Act*[1], `Safe-DL` is designed to bridge the gap between theoretical security concerns and practical implementation. By providing automated risk assessments and interpretable outputs, the framework supports the responsible deployment of AI in sensitive and high-impact environments.

1.1 The Rise of DNNs: the Security Challenge

The pervasive adoption and transformative impact of DL technologies are underscored by significant economic indicators: the global DL market was estimated at USD 96.8 billion in 2024 and is expected to grow at a compound annual growth rate exceeding 31.8% from 2025 to 2030.[2] DNNs, central to this revolution, demonstrate an extraordinary ability to learn complex patterns from vast datasets, achieving high accuracy and efficiency across domains such as medical imaging, autonomous driving, and financial fraud detection. These models now power a growing array of applications that reshape both industries and daily life.

However, as DNNs are increasingly embedded in safety-critical systems, a concerning paradox has emerged: their exceptional performance is accompanied by inherent vulnerabilities to subtle yet potent adversarial manipulations (Yuan et al., 2019; Ren et al., 2020; X. Liu et al., 2020; Khamaiseh et al., 2022). These vulnerabilities allow attackers to craft adversarial examples, which are inputs that appear normal to humans but cause misclassifications in the model. This raises

[1] See: https://artificialintelligenceact.eu/

[2] See: https://www.grandviewresearch.com/industry-analysis/deep-learning-market

fundamental questions about the reliability of AI systems, particularly when deployed in high-risk environments such as autonomous vehicles or clinical diagnostics, where minor prediction errors may lead to severe real-world consequences. Addressing these threats is no longer a technical luxury but a societal and ethical necessity.

Security, therefore, must become a first-class concern in DL system design. As Section 1.2 explores in more detail, adversarial threats compromise not only model performance but also the transparency and accountability required by emerging regulations such as the *AI Act* (Costa et al., 2024; Peng et al., 2024; Kashyap et al., 2024). Understanding these threats is a prerequisite to building trustworthy AI.

1.2 The Problem of Adversarial Attacks in Deep Learning

Adversarial attacks exploit the high dimensionality and sensitivity of DNNs to introduce subtle, often imperceptible perturbations into otherwise legitimate input data. These minor alterations, while indistinguishable from human observers, can cause models to produce severely incorrect or even maliciously manipulated outputs, with potentially critical consequences.

Such vulnerabilities are particularly concerning in safety-critical domains. In medical imaging, a well-crafted perturbation on an X-ray or MRI scan could lead an AI system to misclassify a malignant tumour as benign, delaying essential treatment and putting patients at risk (Ma et al., 2021; Finlayson et al., 2019; Puttagunta et al., 2023). In autonomous vehicles, small changes to road signs or environmental input can deceive perception models, resulting in misinterpretation of traffic signals or obstacles (Badjie et al., 2024; Deng et al., 2020; Girdhar et al., 2023).

These examples illustrate not only the physical harm such attacks may cause but also the broader challenge they pose to public trust and societal acceptance of AI systems. Robust defences are needed to ensure the responsible and secure deployment of these models in high-stakes environments. To better understand what is missing in current solutions, Section 1.3 takes a closer look at their limitations and motivates the need for a more integrated and proactive security framework.

1.3 Gaps in Current Deep Learning Security Approaches

Despite the growing body of research on adversarial threats and defensive strategies, existing approaches to securing DL systems remain fragmented, reactive, and often insufficiently practical. Many defences are narrowly tailored to specific attack types, lacking generalizability or adaptability across threat categories. This results in a patchwork of isolated solutions rather than a cohesive, end-to-end security pipeline. Moreover, these defences are frequently reactive, developed in response to newly discovered attack techniques rather than proactively integrated into the lifecycle of DL model development.

Several research efforts have proposed frameworks or evaluation platforms to address these concerns. Some focus exclusively on adversarial detection (Sun et al., 2020), while others propose

domain-specific pipelines for threat mitigation in areas such as malware classification (D. Li et al., 2021). Other initiatives, such as the MITRE ATLAS[3] knowledge base and Microsoft's Counterfit[4], offer valuable resources for benchmarking and cataloguing attack techniques, but fall short in terms of automated mitigation or integration with user-centric decision workflows.

In most cases, these tools presume expert users and overlook the needs of practitioners unfamiliar with adversarial machine learning or DNN internals. From a usability perspective, many existing frameworks lack accessibility for non-specialists, offer minimal guidance for configuration or defence selection, and fail to provide transparent, auditable output. These limitations conflict directly with the principles of explainability, auditability, and accessibility emphasized by recent regulatory initiatives such as the *AI Act*.

As the use of DL models extends into sensitive and safety-critical domains, security frameworks must be technically robust, understandable, transparent, and aligned with legal and ethical expectations. This dissertation addresses these critical gaps by proposing `Safe-DL`, a modular, user-friendly, and lifecycle-oriented framework for adversarial robustness. While much of the existing literature focuses on evasion and poisoning threats, other privacy-related attacks, such as model extraction, membership inference, and model inversion, remain under-represented in integrated security pipelines. Although these are not yet implemented in the current version of `Safe-DL`, they were considered in the architectural design and are further explored in Section 5.3.2 as promising directions for extending threat coverage.

By unifying threat modelling, attack simulation, risk analysis, and guided defence application, `Safe-DL` enables practitioners, regardless of expertise, to systematically assess and enhance the resilience of DL models. To formalize these goals and guide the development of a practical response to these limitations, Section 1.4 outlines the main objectives of this dissertation.

1.4 Dissertation Objectives

This section defines the central aims that motivated the development of the `Safe-DL` framework, along with the research questions and hypotheses that structure its scientific foundation. Together, these elements guide the architectural design, evaluation methodology, and practical implementation of the system. They also serve to formalize the contribution of this dissertation in addressing real-world challenges in DL security, usability, and regulatory alignment.

1.4.1 General and Specific Objectives

The central objective of this dissertation is to design and develop `Safe-DL`, a modular, systematic, and user-friendly framework for assessing and enhancing the security of DNNs. The framework aims to support practitioners, regardless of their expertise in adversarial machine learning, in identifying, simulating, and mitigating a wide range of adversarial threats. In addition to improving

[3] See: https://atlas.mitre.org/
[4] See: https://github.com/Azure/counterfit

technical robustness, `Safe-DL` emphasizes transparency, explainability, and auditability to support broader societal and regulatory expectations, including those outlined in the *AI Act*.

To structure this overarching goal into actionable components, the work is guided by a series of specific objectives, each corresponding to a key phase of the framework's architecture. These are:

1. **Threat Modelling and Contextualization:** Design an interactive, user-guided questionnaire that allows practitioners to define the context of use, identify relevant attack surfaces, and characterize both the model and dataset. This supports tailored threat modelling per each AI system's specific risk profile and usage domain.

2. **Simulation of Adversarial Attacks:** Implement a diverse set of adversarial attacks, including evasion, data poisoning, and backdoor techniques, ensuring they are customizable, reproducible, and representative of real-world threats encountered in critical domains such as healthcare and autonomous systems.

3. **Risk Assessment and Impact Analysis:** Develop a structured process for evaluating the consequences of simulated attacks through normalized metrics, such as attack success rate, clean accuracy degradation, and class-wise vulnerability. Aggregate these indicators into composite risk scores to support prioritization, enable interpretability, and guide downstream defensive actions.

4. **Automated Defence Configuration and Application:** Design and implement a mechanism that interprets the outputs of prior risk assessments to select and configure appropriate defence strategies automatically. Defensive techniques such as data sanitization, adversarial training, and anomaly detection are applied through a unified interface.

5. **Evaluation of Defence Effectiveness:** Compare and rank candidate defences using composite evaluation metrics that capture mitigation impact, side effects such as accuracy degradation on clean data, and estimated implementation cost. This scoring process is designed to provide actionable guidance aligned with the system's usage context.

6. **Audit-Ready, Reproducible Reporting:** Ensure that all pipeline stages produce structured and human-readable outputs, such as YAML configurations, JSON logs, and Markdown summaries, to support reproducibility, formal documentation, and regulatory compliance.

7. **Support for Future Extensions:** Architect the framework to accommodate future extensions, such as the inclusion of privacy-related attacks, including model extraction, membership inference, and model inversion. Although these are not yet implemented, the system is designed to facilitate their seamless integration in later work.

1.4.2 Research Questions

The development of `Safe-DL` is guided by a set of research questions that reflect key challenges in securing DL models. These questions align with the modular objectives of the framework and serve to orient both design and evaluation efforts:

1. **How can threat modelling be made accessible and effective for non-expert users in the context of deep learning security?** This question targets the need for intuitive and context-aware tools that enable practitioners to identify relevant threats without requiring deep adversarial expertise.

2. **What are the most critical categories of adversarial attacks on DNNs, and how can they be systematically simulated in a controlled and reproducible environment?** This question underpins the implementation of realistic, customizable attack simulations that reflect actual vulnerabilities across domains such as healthcare and autonomous systems.

3. **Which metrics best capture the impact, visibility, and severity of adversarial attacks across different datasets and model architectures?** This question informs the design of comprehensive evaluation criteria that can support both technical decision-making and external auditability.

4. **How can defence strategies be automatically selected and adapted based on prior risk assessments and attack profiles?** This question motivates the development of decision logic that maps specific threats to appropriate countermeasures in a modular and explainable manner.

5. **What is an effective methodology for evaluating and ranking defence mechanisms in terms of mitigation performance, unintended side effects, and implementation cost?** This question supports the design of a composite scoring process to compare candidate defences in a given operational context.

6. **How can reproducibility, explainability, and auditability be ensured throughout the adversarial robustness pipeline to support regulatory compliance and trustworthy AI deployment?** This question addresses the overarching requirement of producing traceable, interpretable, and reusable artifacts that align with the principles of responsible AI.

1.4.3 Hypotheses

To complement the objectives and guide the empirical validation of `Safe-DL`, the following hypotheses are formulated. Each one corresponds to a core capability or design principle of the framework and will be addressed through the experiments, results, and critical discussion provided in later chapters:

1. **`Safe-DL` enables reliable identification of adversarial vulnerabilities in DL models across a range of threat types.** This hypothesis evaluates whether the framework can simulate and expose vulnerabilities effectively, supporting the claim that adversarial robustness requires contextual and multi-faceted analysis.

2. **The automated defence selection and evaluation mechanism in `Safe-DL` produces mitigation strategies that balance effectiveness, side effects, and implementation cost.** This hypothesis focuses on the practical viability of the pipeline's defence recommendation and scoring system, assessing whether it can support informed decision-making across different operational contexts.

3. **The framework's configuration and reporting architecture provide a usable, auditable, and extensible basis for adversarial risk assessments, even for non-expert users.** This hypothesis addresses the accessibility and transparency of the system, particularly its capacity to serve practitioners with limited experience in adversarial machine learning while still complying with emerging requirements for trustworthy AI.

Together, the objectives, research questions, and hypotheses outlined above form the scientific backbone of this dissertation. Before diving into the technical architecture and experimental validation, it is worth briefly outlining the concrete contributions that emerged from this process. These are summarized in Section 1.5, which highlights the key results achieved throughout the development of the `Safe-DL` framework.

1.5 Contributions of the Dissertation

This dissertation delivers several key contributions to the field of deep-learning security, centred on the development of `Safe-DL`, a modular and extensible framework for systematically assessing and mitigating adversarial threats. The framework is designed to satisfy both technical robustness and practical usability, aligning with the needs of researchers, practitioners, and regulatory bodies.

Design and Implementation of the `Safe-DL` Framework. A modular, end-to-end system composed of six interconnected modules that together cover the entire adversarial security lifecycle, from threat modelling and attack simulation to defence selection and final reporting.

Lifecycle-Oriented Architecture. The framework enables a seamless, reproducible pipeline across all stages of robustness evaluation, replacing the fragmented, single-stage focus typical of many existing tools.

Accessibility for Non-Experts. Guided configuration via interactive questionnaires, default profiles and context-aware parameter recommendations lowers the barrier for practitioners who lack deep expertise in adversarial machine learning.

Auditability and Reproducibility. All configurations and outputs are stored in structured formats (YAML, JSON, Markdown), supporting versioning, formal documentation, and audit readiness that are in line with emerging AI-governance requirements.

Actionable and Interpretable Insights. Beyond mere attack detection, `Safe-DL` generates composite risk scores and ranked mitigation strategies derived from multi-metric analyses, enabling informed, context-aware decision-making.

These contributions translate the objectives and hypotheses defined earlier into tangible results that underpin the remainder of this dissertation. The overall organisation of the document and how each chapter develops these contributions in detail is outlined in Section 1.6.

1.6 Structure of the Dissertation

The remainder of this dissertation is organised into four main chapters, followed by a set of appendices and supporting materials. Each chapter contributes to addressing the research questions and validating the hypotheses introduced earlier in Section 1.4.2 and 1.4.3.

- **Chapter 2 — Theoretical Background and State of the Art**: Presents a structured literature review covering DNNs, their development lifecycle, known adversarial threats, defence strategies, and existing evaluation frameworks. This chapter provides the foundation for the architectural and methodological decisions behind `Safe-DL`.
- **Chapter 3 — Safe-DL Framework: Architecture and Modules**: Introduces the design and implementation of the `Safe-DL` framework. Each of its six modules—threat modelling, attack simulation, risk analysis, defence application, evaluation, and reporting—is described in detail, along with its internal logic and technical contributions.
- **Chapter 4 — Experimental Validation and Framework Discussion**: Presents the experimental setup and results of applying `Safe-DL` to a representative use case. The framework is exercised end-to-end to demonstrate its effectiveness, usability, and regulatory alignment. The chapter concludes with a critical discussion of strengths, limitations, and comparisons with existing tools.
- **Chapter 5 — Conclusion and Future Work**: Summarises the contributions of the dissertation, reflects on the lessons learned, and outlines future directions for expanding the framework, improving its components, and supporting broader adoption in real-world scenarios.

In addition to the main chapters, the dissertation includes appendices with supporting material and selected outputs from the execution of the `Safe-DL` pipeline. The complete source code of the framework is available in the project's public repository[5].

1.7 Chapter Summary

This chapter introduced the core motivations, challenges, and goals that underpin this dissertation. It began by highlighting the transformative role of DNNs in a wide range of fields, alongside

[5] See: https://github.com/th0rz05/safe-dl-framework

their growing exposure to adversarial threats. The discussion then framed these vulnerabilities not only as technical challenges but also as issues with regulatory, societal, and ethical implications, particularly in light of emerging standards such as the *AI Act*. This contextual grounding set the stage for the introduction of `Safe-DL`, a security-focused framework designed to improve both the robustness and transparency of DL systems.

Following this, the chapter defined the primary research objectives of the dissertation and articulated a set of specific goals corresponding to each phase of the `Safe-DL` framework. These were further complemented by guiding research questions and hypotheses, which together provide a clear structure for the theoretical inquiry and experimental validation developed in later chapters. The contributions of the work were also summarised, emphasising not only the functional capabilities of the framework but also its alignment with usability, explainability, and auditability requirements that are increasingly vital in modern AI development.

With the motivations, goals, and scope of the work now clearly established, the next step is to explore the theoretical foundations upon which `Safe-DL` is built. Chapter 2 provides this foundation by reviewing key concepts in DNN security, surveying existing adversarial threats and defence strategies, and identifying critical requirements that inform the design of the proposed framework.

Chapter 2

Theoretical Background and State of the Art

This chapter establishes the theoretical foundation and contextual background needed to address the security challenges explored in this dissertation. Building upon the problem formulation and motivation introduced in Chapter 1, it outlines the methodology adopted to conduct a rigorous and reproducible literature review.

The discussion then shifts to core architectural and training concepts of DNNs, emphasising the stages most vulnerable to adversarial interference. A structured taxonomy of adversarial threats follows, organised by attack modality and system vulnerability, providing a clear landscape of current challenges.

Subsequently, the chapter examines defensive strategies proposed in the literature, highlighting their technical merits and practical limitations. This review is complemented by an analysis of existing security frameworks for DL, with particular attention to their gaps in usability and lifecycle integration.

The chapter concludes by consolidating insights from the literature into a set of technical and usability-driven requirements that directly inform the architectural design of `Safe-DL`. These requirements serve as a bridge between theoretical understanding and the framework proposed in the following chapters.

2.1 Literature Review Methodology

To ground this dissertation in recent and high-quality research, a systematic literature review was conducted following principles inspired by the *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA)[1]. This methodology ensures transparency, reproducibility, and a structured approach to identifying and selecting relevant studies in the field of adversarial DL.

[1] See: https://www.prisma-statement.org/

The search focused on three leading digital libraries in computer science and engineering: IEEE Xplore[2], Scopus[3], and the ACM Digital Library[4]. These databases were selected for their extensive peer-reviewed collections in machine learning, cybersecurity, and DL research. Articles were filtered based on relevance, recency (2018 onward), accessibility, and publication type, with only peer-reviewed journal articles, conference papers, or book chapters written in English being considered.

Semantic search capabilities of the AI-assisted Consensus[5] platform were also employed to broaden coverage and capture emerging research trends. Additionally, several foundational studies were manually included based on their relevance or supervisor recommendation.

The complete search strategy, inclusion and exclusion criteria, screening steps, and PRISMA flow diagram are fully documented in Appendix A.

Having defined how the literature review was conducted, attention now turns to the architectural foundations and lifecycle dynamics of DL systems, which are explored in more detail in Section 2.2.

2.2 DNNs and Security-Relevant Concepts

DL models rely on architectural and training choices that directly influence their exposure to adversarial vulnerabilities. These design factors, ranging from gradient flow and optimisation strategies to preprocessing pipelines, shape the potential attack surfaces that malicious actors may exploit. A clear understanding of these technical foundations is essential for designing secure and interpretable systems capable of resisting sophisticated threats.

2.2.1 Core Architecture and Training Pipeline

A standard DNN comprises multiple layers of interconnected neurons, typically organised into input, hidden, and output layers. These models learn by minimising a loss function using stochastic gradient descent or one of its variants. Throughout the training process, model parameters are updated iteratively by computing the gradients of the loss concerning the weights, using backpropagation (X. Liu et al., 2020; Yuan et al., 2019).

Several architectural and training design choices influence the robustness and vulnerability of DNNs. These include the type of activation functions, the structure and depth of the network, regularisation techniques such as dropout, and the optimisation algorithm employed. For example, convolutional neural networks (CNNs), commonly used in image-related tasks, exploit spatial locality and parameter sharing, enabling efficient learning of hierarchical representations while also introducing structural patterns that can be targeted by specific attack vectors (Y. Zhou et al., 2019; Akhtar et al., 2021).

[2] https://ieeexplore.ieee.org/
[3] https://www.scopus.com/
[4] https://dl.acm.org/
[5] https://consensus.app/

A solid understanding of these components is crucial for performance tuning and anticipating how adversaries might exploit weaknesses embedded in architectural decisions or training routines.

2.2.2 Security-Sensitive Stages in the DNN Lifecycle

To develop effective security mechanisms, it is crucial to understand which stages of the DL lifecycle are most susceptible to adversarial threats. Vulnerabilities can emerge at multiple points, including data ingestion, model training, and post-deployment inference. Each of these stages presents unique opportunities for exploitation. This subsection identifies and characterises these critical stages, providing a structured foundation for designing targeted defences in later chapters.

Input Preprocessing. The input preprocessing stage is a common attack surface where adversaries introduce subtle perturbations that are imperceptible to humans but can drastically alter model predictions. Preprocessing operations such as normalisation, resizing, and feature extraction may inadvertently amplify or fail to suppress these adversarial perturbations (X. Liu et al., 2020; Thangaraju et al., 2022).

Training Phase. During the training phase, the model becomes particularly sensitive to data quality. Poisoning attacks such as label flipping, outlier injection, or backdoor insertion can compromise model integrity by embedding malicious patterns into the training data. Even a small proportion of manipulated samples may induce significant degradation in generalisation (Khamaiseh et al., 2022; Costa et al., 2024).

Gradient Flow and Optimisation. Another critical point of vulnerability lies in the gradient flow and optimisation process. Many white-box attacks exploit gradient-based information to craft adversarial inputs. Methods like the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) compute optimal perturbations using the model's gradients with respect to inputs. In response, some defences attempt to distort or obscure gradient information to mitigate such vulnerabilities (Ren et al., 2020; Chen et al., 2021).

Model Update and Fine-Tuning. Further risks emerge during the model update and fine-tuning phases, especially in transfer or continual learning contexts. Attackers can introduce poisoned data during retraining or adaptation, compromising downstream performance or embedding latent malicious behaviours that are difficult to detect (Machooka et al., 2023; M. Li et al., 2021).

Inference and Post-Processing. Finally, models are exposed to inputs crafted during inference and post-processing test time. These adversarial examples exploit the learned decision boundaries and typically require no access to the training data. Because inference is often a deterministic and continuous process, models remain vulnerable in production environments unless protected by proactive defence mechanisms (C. Wang et al., 2022; Xue et al., 2020).

Understanding the vulnerabilities inherent to each stage of the DNN lifecycle provides the basis for the lifecycle-aware design of `Safe-DL`, which addresses these points of weakness through modular threat detection, simulation, and defence strategies.

2.2.3 Implications for Robustness and Explainability

The complexity and non-linearity inherent to DNNs are key enablers of their strong performance across tasks. However, these properties often obscure the model's internal decision-making processes, making them difficult to interpret, audit, or defend. This opacity is particularly problematic in safety-critical domains, where understanding model behaviour is crucial for trust and accountability (Ximeng Liu et al., 2021; Arshad et al., 2023).

In addition, a large number of parameters and the tendency to overfit training data can expose DNNs to memorisation-based threats, including membership inference and model inversion attacks (Yanjie Li et al., 2024; S. Zhou; C. Liu, et al., 2022). These attacks exploit the model's capacity to retain detailed information about its training data, raising concerns over privacy and data leakage.

While DNNs offer remarkable representational capabilities, their architectural and training characteristics introduce critical robustness and explainability challenges. Understanding these limitations is a prerequisite for designing effective and targeted security mechanisms—a theme further explored in the subsequent sections.

2.2.4 Applicability Beyond Vision

While this dissertation centres on image classification, the security vulnerabilities it addresses arise from architectural and optimisation characteristics that are shared across application domains. These properties, such as gradient-based optimisation and high model capacity, underpin similar adversarial behaviours in other fields. For example, sentiment analysis has demonstrated adversarial perturbations for NLP (X. Liu et al., 2020) and malware detection pipelines (Ximeng Liu et al., 2021).

Such cross-domain findings highlight the need for security mechanisms that generalise beyond specific data modalities. In alignment with this, the modular architecture of `Safe-DL` was deliberately designed to support future extensions into non-vision contexts.

The architectural and lifecycle characteristics discussed above reveal how multiple stages of a DNN pipeline introduce distinct opportunities for adversarial exploitation. These vulnerabilities are not isolated but reflect systemic properties that adversaries can target through various strategies. Building upon this technical foundation, Section 2.3 presents a structured overview of the main categories of adversarial threats, highlighting how each exploits different stages and mechanisms within DL systems.

2.3 Attack Strategies in the Literature

As the deployment of DNNs expands into safety-critical and adversarially exposed domains, understanding how these systems can be compromised has become a central concern in the security

of AI. Over the past decade, researchers have identified various attack strategies that exploit different stages of the model lifecycle—from training-time manipulation to inference-time evasion and post-deployment privacy leakage. These threats are not isolated anomalies but systemic vulnerabilities that emerge from the architectural and statistical properties of DNNs. Recognising their scope and internal logic is crucial for designing comprehensive defence strategies, which requires a structured understanding of the adversarial landscape observed in the literature.

2.3.1 Overview and Taxonomy of Attacks

The widespread deployment of DNNs in critical domains has made them a prime target for adversarial attacks, which exploit the inherent vulnerabilities of these models to compromise their reliability and security (Akhtar et al., 2021; Yuan et al., 2019). These attacks vary significantly in scope and mechanism, ranging from subtle perturbations at inference time to malicious data manipulation during training. To understand and mitigate such threats, the research community has proposed taxonomies based on multiple axes: attack timing, adversary knowledge, and attacker intent (S. Zhou; C. Liu, et al., 2022; M. Li et al., 2021).

From the machine learning pipeline perspective, attacks are classified into training and inference time. Training time attacks include data poisoning, where malicious samples are injected into the training set to corrupt the learned decision boundaries, and backdoor attacks, which embed hidden triggers to manipulate model behaviour during deployment (Xue et al., 2020; Ximeng Liu et al., 2021). Inference-time attacks, conversely, involve crafting adversarial examples that cause models to misclassify inputs by applying imperceptible perturbations (Ren et al., 2020; C. Wang et al., 2022).

Another crucial dimension in categorising adversarial threats is the attacker's access level to the model. In white-box settings, the adversary fully knows the model's architecture, parameters, and training data. This allows for highly effective gradient-based attacks such as FGSM, PGD, and Carlini-Wagner (C&W) methods (Thangaraju et al., 2022; Yuan et al., 2019). Conversely, black-box attacks assume the adversary only has query access to the model's outputs. These attacks rely on strategies such as transferability, where adversarial examples generated for a substitute model successfully fool the target, or on building surrogate models via query synthesis (C. Wang et al., 2022; Machooka et al., 2023).

Adversarial objectives vary, from targeted attacks that aim to force a specific misclassification to untargeted attacks that simply degrade model accuracy (X. Liu et al., 2020). Beyond classification evasion, other attack classes include model stealing, where an attacker replicates a model's behaviour to clone its functionality; model inversion and membership inference, which aim to recover sensitive training data and compromise privacy (Peng et al., 2024; Xue et al., 2020).

Some surveys have proposed unifying frameworks that align these threats with classical security goals. For instance, poisoning attacks affect model integrity, inversion and stealing breach confidentiality, and adversarial examples may compromise availability in mission-critical applications (Xue et al., 2020; Machooka et al., 2023).

This taxonomy maps the diverse threat landscape of adversarial machine learning and provides a foundation for developing and evaluating effective defence mechanisms in subsequent sections.

2.3.2 Data Poisoning Attacks

Data poisoning attacks aim to subvert the learning process of DNNs by injecting malicious samples into the training dataset. Rather than altering the model post-deployment, these attacks exploit the model's sensitivity to data quality, embedding vulnerabilities that manifest during inference. Poisoning is particularly dangerous when training data is aggregated from external or untrusted sources, such as public datasets or federated learning scenarios.

Label Flipping Attacks. These attacks modify the labels of a subset of training examples while keeping the input features unchanged. By introducing incorrect associations, the model is guided to learn misleading decision boundaries. Their effectiveness is influenced by both the fraction of altered labels and the flipping strategy—random flipping distributes errors uniformly, whereas targeted flipping focuses on specific misclassifications (Dang et al., 2020; C. Hu et al., 2020; Khamaiseh et al., 2022).

Clean-Label Attacks. In clean-label poisoning, the attacker cannot modify labels and instead perturbs input features to embed malicious behaviour. Despite having correct labels, these crafted samples distort the decision boundary in a way that causes specific misclassifications at test time. These attacks are stealthy and realistic in scenarios involving crowdsourced or third-party data (Huang et al., 2021; Costa et al., 2024; Priya et al., 2023).

The objectives behind poisoning vary, ranging from broad degradation of model performance to targeted misclassification of individual samples. Standard evaluation metrics include clean accuracy drop, attack success rate, and the perceptual or statistical indistinguishability of the poisoned data (Akhtar et al., 2021; Ren et al., 2020; Machooka et al., 2023). Alarmingly, in many cases, only a small fraction of poisoned inputs are needed to induce large-scale model failure.

Poisoning feasibility depends heavily on the attacker's access to the data pipeline. In direct poisoning, the adversary can inject samples into the dataset before training. In indirect scenarios—such as pre-trained model supply chains—attacks can be performed upstream during transfer learning or fine-tuning, allowing backdoors to propagate silently into downstream applications (M. Li et al., 2021; Arshad et al., 2023).

Detection remains a significant challenge due to the similarity between poisoned and benign inputs. Various defences have been proposed, including clustering, spectral methods, and influence functions (Peng et al., 2024; Chen et al., 2021), yet more stealthy poisoning strategies can circumvent these mechanisms. As such, robust data validation protocols are increasingly considered essential components of secure model development pipelines.

Data poisoning attacks highlight a critical vulnerability in the training phase of DNNs. By injecting a relatively small number of malicious samples, adversaries can induce significant misbehaviour, often without visible traces. This subtlety, combined with the increasing reliance on large, third-party, or crowdsourced datasets, makes poisoning a particularly pressing threat to the integrity and reliability of modern machine learning systems.

2.3.3 Backdoor Attacks

Backdoor attacks are a stealthy and targeted subclass of data poisoning strategies. Rather than degrading a model's overall performance, these attacks implant malicious behaviours that activate only in the presence of specific input patterns known as *triggers*. A typical pipeline involves three steps: first, the attacker designs or learns a trigger pattern; second, they poison part of the training dataset by embedding the trigger into selected samples (optionally relabelling them); and finally, the model is trained on the mixed dataset, learning to associate the trigger with a target class. When the trigger is absent, the model behaves normally; when it is present, it is misled into producing attacker-controlled outputs. This deceptive behaviour makes backdoor attacks particularly dangerous in real-world systems, where models are often assumed to be reliable and trustworthy (Gao; Doan, et al., 2020; W. Guo et al., 2022).

Static Trigger Backdoors. These attacks rely on predefined, manually crafted patterns—such as coloured squares, geometric shapes, or overlayed symbols—inserted into a subset of the training images. One of the earliest examples, BadNets, used a white square placed in a fixed location to consistently redirect predictions to a chosen target class (W. Guo et al., 2022). Variants of static triggers include: visible patches, which are distinguishable but often ignored by training pipelines; blended patterns, which mix the trigger with the input image at low intensity to reduce perceptibility (Gao; Doan, et al., 2020); invisible perturbations, which apply subtle pixel-level noise optimised to be imperceptible to the human eye (Yudong Li et al., 2023); and physical-world triggers, such as stickers applied to road signs, maintain their effect under real-world transformations (Dang et al., 2020; W. Guo et al., 2022). These static methods are simple to implement and highly effective when the attacker controls part of the training data.

Learned Trigger Backdoors. Instead of designing the trigger manually, some attacks optimise the trigger pattern during training, often jointly with model weights. This results in imperceptible or subtle triggers that are harder to detect. Such methods frequently use adversarial optimisation or gradient-based learning to create inputs that embed the trigger while maintaining clean-labels (Gao; Doan, et al., 2020; Yudong Li et al., 2023). These attacks are instrumental in constrained threat models where the adversary cannot alter labels. The resulting triggers can be minimal in size, dynamically placed, or even encoded as noise invisible to the human eye, making them highly stealthy and resilient to basic filtering defences.

Backdoor attacks can follow either a *corrupted-label* or *clean-label* strategy. In corrupted-label scenarios, the attacker injects trigger-bearing samples with incorrect target labels to guide the model's behaviour. This setting is effective but more detectable through data auditing or outlier detection. In contrast, clean-label backdoors preserve the correct labels, making poisoned samples appear indistinguishable from legitimate data. Clean-label attacks often require more sophisticated trigger learning techniques but are much harder to detect in practice (W. Guo et al., 2022).

Backdoor attacks focus more on scope than general-purpose data poisoning and are less disruptive to clean performance. While traditional poisoning aims to broadly degrade accuracy or shift decision boundaries globally, backdoors are designed to keep normal behaviour intact and

activate only under specific conditions. This distinction makes them particularly covert and persistent across model updates or transfer learning (C. Hu et al., 2020; Yudong Li et al., 2023).

Backdoor attacks illustrate the dangers of assuming training data is trustworthy. Their ability to implant hidden logic into models without impacting visible behaviour poses a serious threat in any setting where data is collected from external or unverified sources. Ensuring the integrity of training pipelines is, therefore, a critical component of any secure machine learning workflow.

2.3.4 Evasion Attacks

Evasion attacks are among the most extensively studied forms of adversarial threats in DNNs. These attacks occur at inference time and aim to mislead the model by introducing carefully crafted perturbations to input data without modifying the model itself. While such perturbations are often imperceptible to the human eye, they can drastically alter a model's output, exposing critical vulnerabilities in seemingly robust systems (Ren et al., 2020; S. Zhou; C. Liu, et al., 2022).

FGSM attack applies a single-step perturbation in the direction of the loss gradient with respect to the input. The formulation is given in Equation (2.1).

$$\mathbf{x}^{\text{adv}} = \mathbf{x} + \varepsilon \cdot \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)), \tag{2.1}$$

Here, ε defines the perturbation magnitude. Even this simple gradient-based modification can suffice to deceive DNNs, effectively exposing structural vulnerabilities in standard training pipelines (Yuan et al., 2019; X. Liu et al., 2020).

PGD attack builds upon FGSM by applying it iteratively and projecting each perturbed example back into the allowed ε-ball around the original input, as shown in Equation (2.2).

$$\mathbf{x}_{t+1} = \Pi_{\mathscr{B}_\varepsilon(\mathbf{x})} \left(\mathbf{x}_t + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}_t} J(\theta, \mathbf{x}_t, y)) \right), \tag{2.2}$$

In this formulation, Π denotes the projection operator onto the ε-ball $\mathscr{B}_\varepsilon(\mathbf{x})$. PGD is widely regarded as a powerful first-order adversary and is frequently used as a benchmark for evaluating model robustness (Akhtar et al., 2021; Machooka et al., 2023).

C&W attack formulates the adversarial objective as a constrained optimisation problem that minimises perturbation while enforcing misclassification. This method bypasses many existing defences, particularly those relying on defensive distillation or gradient obfuscation (X. Liu et al., 2020; Thangaraju et al., 2022).

DeepFool attack is an untargeted attack that iteratively computes the minimal perturbation required to move a sample across the decision boundary. It assumes the classifier's local linearity and effectively identifies vulnerable regions near class boundaries (Yuan et al., 2019).

Natural Evolution Strategies (NES) attack is a black-box method that estimates the gradient via random perturbations and output sampling. Since it requires only access to confidence scores, it is well-suited to settings where internal model details are not available (S. Zhou; C. Liu, et al., 2022; C. Wang et al., 2022).

Simultaneous Perturbation Stochastic Approximation (SPSA) attack follows a similar philosophy but introduces simultaneous perturbations across all input dimensions. It scales nicely to high-dimensional inputs and operates under limited information settings (S. Zhou; C. Liu, et al., 2022; C. Wang et al., 2022).

Transfer-based attacks exploit the observation that adversarial examples often generalise across models. By generating perturbations on a surrogate model, attackers can successfully mislead other unseen models with similar architectures or training objectives (S. Zhou; C. Liu, et al., 2022; Machooka et al., 2023).

Boundary attacks operate in a decision-based black-box setting, starting from a heavily perturbed input already misclassified and iteratively reducing the perturbation while remaining within the adversarial region. This method relies only on final prediction outputs and is effective even without access to gradients or confidence scores (Brendel et al., 2018).

Evasion attacks are frequently categorised according to the level of access granted to the adversary. *White-box attacks* assume complete model knowledge, including architecture, parameters, and gradients—as in FGSM, PGD, C&W, and DeepFool. In contrast, *black-box attacks* operate without internal access, leveraging only input-output behaviour, as exemplified by NES, SPSA, transfer-based, and boundary methods (X. Liu et al., 2020; Ren et al., 2020; Brendel et al., 2018).

Overall, evasion attacks underscore fundamental fragilities in DNNs. Their success often requires minimal perturbations and no privileged access, making them a pervasive threat in real-world deployments, particularly in safety-critical domains such as healthcare, finance, and autonomous systems.

2.3.5 Model Extraction Attacks

Model extraction attacks, also known as model stealing attacks, target the replication of a model's functionality by leveraging query access to its output. These attacks are particularly dangerous in Machine Learning as a Service (MLaaS) settings, where proprietary models are exposed through Application Programming Interfaces (APIs). By systematically querying the model and observing its predictions, an adversary can reconstruct a high-fidelity surrogate model without access to internal parameters, architecture, or training data. The stolen model may be used for downstream attacks, re-deployment, or intellectual property theft.

Knockoff Nets. One of the earliest and most influential demonstrations of model extraction was proposed by Orekondy et al. (2018). In this setup, a substitute model is trained by querying a victim model with inputs sampled from a distribution that loosely resembles the original training data. Despite limited knowledge of the model's design or data, the resulting knockoff model achieved competitive performance, showing that high-fidelity replication is possible under practical constraints.

Query-based Extraction. A more general strategy involves crafting large sets of inputs and collecting the model's corresponding outputs to train a surrogate. These inputs may be seeded from a natural data distribution, synthetically generated via noise or optimisation, or selected adaptively

based on previous queries. As shown by Jiang et al. (2024), adaptive query strategies significantly improve extraction efficiency by prioritising informative regions of the input space.

These attacks' effectiveness highly depends on the information exposed through the model's interface. When APIs return confidence scores or logits instead of just class labels, they provide a richer learning signal that accelerates the extraction process (Juuti et al., 2019). Even limited black-box access can thus leak enough information to reconstruct complex decision boundaries.

Model extraction attacks expose a fundamental vulnerability in deployed machine learning systems: external access to predictions alone may be enough to compromise proprietary models. Their feasibility under realistic threat models makes them a pressing concern, especially in commercial or security-sensitive deployments.

2.3.6 Membership Inference Attacks

Membership inference attacks aim to determine whether a specific data point was used to train a target model. They exploit statistical discrepancies in the model's behaviour, such as differences in confidence scores, prediction entropy, or loss, when responding to inputs seen during training versus previously unseen data. These subtle variations arise from imperfect generalisation, particularly in overfitted models that memorise aspects of the training set (J. Ye et al., 2022).

These attacks pose a significant privacy risk in domains where training data may encode sensitive or personally identifiable information, such as healthcare records, financial transactions, or user behaviour logs. In such contexts, the ability to infer an individual's presence in the training dataset can result in serious privacy breaches, regulatory violations, and reputational damage (H. Hu et al., 2022).

The **Shadow Model Attack** is one of the most widely studied approaches. In this strategy, the adversary first trains a separate model, referred to as the shadow model, on a dataset similar to the one assumed to be used by the target. This shadow model is a proxy, allowing the attacker to observe how training versus non-training samples behave. Using this information, the attacker trains an inference model to classify whether a given sample is likely part of the target's training set (Shuvo et al., 2020; H. Hu et al., 2022).

Depending on the attacker's level of access, membership inference attacks can operate in black-box or white-box scenarios. In black-box settings, only the model's outputs, such as class probabilities or predicted labels, are accessible. In white-box settings, attackers can exploit internal gradients, weight norms, or training statistics to improve inference accuracy (L. Hu et al., 2023; J. Ye et al., 2022). More adaptive attacks dynamically adjust their strategy depending on the observed responses and dataset characteristics, often combining multiple signals to boost robustness (Shuvo et al., 2020).

The success of membership inference is influenced by a variety of factors. Models with high generalisation gaps, where training accuracy significantly exceeds test accuracy, are particularly vulnerable. Similarly, deep architectures with high capacity are more likely to overfit and leak information if not properly regularized (J. Ye et al., 2022). Dataset properties such as small size,

class imbalance, or skewed distributions can also exacerbate the risk, especially when samples from underrepresented classes are easier to distinguish (L. Hu et al., 2023).

Membership inference attacks reveal how overfitting and poor generalisation can directly translate into privacy risks. Their low access requirements, particularly in black-box settings, make them highly applicable in real-world systems. Importantly, these attacks pose serious concerns under data protection regulations such as the General Data Protection Regulation (GDPR)[6], which explicitly recognises membership inference as a form of personal data exposure (H. Hu et al., 2022). As such, they represent a tangible threat to data confidentiality in DL deployments.

2.3.7 Model Inversion Attacks

Model inversion attacks aim to reconstruct sensitive input data by exploiting access to a trained model, often revealing personal or proprietary information about individuals in the training set. These attacks raise significant privacy concerns in domains such as medical imaging, facial recognition, and biometric authentication, where training data is often private and identity-linked.

White-box Model Inversion Attacks. Traditional inversion methods assume full access to the model's internal parameters, such as weights, gradients, or confidence scores. By maximising the output confidence for a given class, attackers can iteratively optimise synthetic inputs that closely resemble representative training examples. These approaches expose how overfitted or overly confident models may unintentionally encode training set information.

Label-Only and Black-box Inversion Attacks. More recent techniques relax the access assumptions, demonstrating that inversion is feasible even in constrained settings. In label-only attacks, such as those proposed by Zhu et al. (2023), the model reveals only predicted class labels without probabilities or gradients. Nonetheless, attackers can leverage generative priors and iterative refinement to synthesise realistic inputs. The C2FMI technique (Z. Ye et al., 2024) pushes this further, using a coarse-to-fine strategy to progressively refine input reconstructions, achieving strong performance even under black-box constraints.

Inversion via Adversarial Examples. A novel perspective combines adversarial perturbation techniques with inversion objectives. As shown by S. Zhou; Zhu, et al. (2024), adversarial examples can help navigate toward decision boundaries that reveal class-specific features. Blending evasion mechanisms with reconstruction goals enhances the fidelity of inverted samples, bridging two major threat categories.

GAN-Based Inversion Attacks. Generative adversarial networks have also been leveraged to support or resist model inversion. On the offensive side, attackers use pre-trained Generative Adversarial Networks (GANs) to guide reconstructions that align with data distribution priors. Defensively, Gong et al. (2023) propose embedding GANs into the model pipeline to distort inversion gradients and obfuscate reconstruction paths without degrading overall model utility.

Model inversion attacks reveal how even limited access to a model's outputs can compromise the confidentiality of training data. Their ability to reconstruct sensitive features from output

[6]GDPR is a European Union regulation that governs the processing of personal data and aims to protect the privacy and rights of individuals. Official text available at: https://gdpr-info.eu/.

signals underscores the urgent need for privacy-aware model design, especially in high-stakes applications such as healthcare or biometrics. These attacks may also constitute a breach of data protection regulations such as the GDPR, which defines the reconstruction of identifiable information from learned models as a form of personal data exposure.

The breadth and sophistication of attack strategies outlined above underscore the multifaceted nature of adversarial threats in DNNs. These attacks exploit vulnerabilities across different stages of the model lifecycle, from training-time manipulation and inference-time evasion to post-deployment privacy leakage. Mitigating such threats requires diverse defensive techniques, each tailored to specific adversarial goals and assumptions. In Section 2.4, we explore defence strategies proposed in the literature, examining their theoretical underpinnings, practical effectiveness, and limitations across various threat scenarios.

2.4 Defence Strategies in the Literature

The increasing sophistication of adversarial attacks has motivated an equally diverse set of defence strategies to preserve the reliability and security of DL models. These mechanisms seek to counteract vulnerabilities exploited at various stages of the machine learning pipeline, ranging from training-time manipulations to inference-time exploits, by improving robustness, detecting anomalies, or preserving data privacy.

As explored in previous sections, the nature and goals of adversarial threats can vary significantly. Accordingly, defences must be carefully tailored to the attack vector in question. Some are designed to proactively harden models against manipulation, while others are reactive, aiming to detect or mitigate malicious behaviour after it has occurred. Each approach involves trade-offs between robustness, computational cost, interpretability, and performance.

In this section, we explore the landscape of defence techniques in the literature, organising them according to the type of threat they are intended to counter. This structure allows for a more precise analysis of how different strategies map to distinct adversarial scenarios and how they contribute to a defence-in-depth approach for securing DNNs.

2.4.1 Overview and Taxonomy of Defences

The emergence of diverse adversarial threats in DL has catalysed the development of a wide array of defensive strategies, each designed to address specific vulnerabilities across the machine learning pipeline. To enable systematic analysis, these defences can be categorised along multiple dimensions, such as the type of threat they target, the stage of the pipeline they operate on, and their temporal nature—whether preventive or reactive (Ren et al., 2020; Akhtar et al., 2021; S. Zhou; C. Liu, et al., 2022).

By attack type, data poisoning defences attempt to sanitise training data or reduce sensitivity to poisoned samples (Dang et al., 2020; C. Hu et al., 2020). Backdoor defences aim to detect or mitigate hidden triggers embedded during training (Gao; Doan, et al., 2020; W. Guo et al., 2022;

Yudong Li et al., 2023). Evasion defences focus on enhancing model robustness during inference, often via adversarial training or certified guarantees (Ren et al., 2020; Akhtar et al., 2021). Privacy-related defences, such as those countering membership inference or model inversion, typically rely on differential privacy (DP) or access restrictions (H. Hu et al., 2022; Zhu et al., 2023; Gong et al., 2023).

Another axis of classification concerns the level at which the defence is applied. Data-level defences are applied before model training and involve manipulations to the dataset, such as sanitisation, outlier removal, or noise injection (Dang et al., 2020; S. Zhou; C. Liu, et al., 2022). Model-level defences operate during training by modifying the architecture or loss function to encourage robustness. These include robust optimisation, regularisation, or dropout techniques (Akhtar et al., 2021; L. Hu et al., 2023). Post-training defences are employed after the model has been deployed and focus on runtime protection mechanisms such as access control, input filtering, or response perturbation (S. Zhou; C. Liu, et al., 2022; H. Hu et al., 2022).

Defences can also be distinguished by their timing. Proactive defences are integrated into the training or deployment process to pre-empt adversarial behaviour. These include strategies like adversarial training, DP, or robust optimisation frameworks (Ren et al., 2020; Akhtar et al., 2021). Reactive defences, in contrast, aim to detect and respond to adversarial activity post-deployment. Examples include anomaly detection, clustering-based analysis, and saliency map inspection (Gao; Doan, et al., 2020; W. Guo et al., 2022; L. Hu et al., 2023).

Despite their importance, many defence strategies come with inherent trade-offs. Enhancing robustness often entails sacrificing clean data accuracy, increasing computational overhead, or reducing model interpretability. Moreover, some defences may fail under adaptive adversaries who explicitly account for the presence of protection mechanisms during attack design (S. Zhou; C. Liu, et al., 2022; Akhtar et al., 2021).

These dimensions provide a conceptual foundation for categorising and comparing defence strategies. In the following subsections, we detail the most relevant defences for each adversarial threat class.

2.4.2 Defences Against Data Poisoning Attacks

Defending against data poisoning attacks is critical to ensuring the integrity of DNNs. These defences aim to detect, mitigate, or render ineffective the presence of malicious samples during the training phase.

Data Cleaning. These techniques identify and remove anomalous or suspicious training samples that may result from poisoning. Many approaches rely on clustering, distance metrics, or influence estimations to flag harmful data. For example, consistency scores and model influence assessments can filter out likely poisoned examples before training (Dang et al., 2020; C. Hu et al., 2020).

Per-Class Monitoring. This strategy observes class-wise metrics such as accuracy or gradient statistics to identify abnormal patterns introduced by label flipping. If a specific class's

performance or loss distribution diverges significantly from expectations, it may indicate targeted poisoning (Huang et al., 2021).

Robust Loss Functions. These are designed to diminish the impact of poisoned data during optimisation. By being less sensitive to outliers, robust loss functions—such as bounded losses or re-weighting schemes—help the model resist the influence of mislabelled or adversarial samples (W. Guo et al., 2022).

Differentially Private Training. DP introduces calibrated noise into the training process, limiting the influence of any single sample. Initially designed for privacy preservation, techniques such as Differentially Private SGD have shown additional benefits in mitigating poisoning, especially when applied during gradient updates (Yudong Li et al., 2023; W. Guo et al., 2022).

Provenance Tracking. By attaching metadata to training samples, such as source information, timestamps, or transformation history, provenance tracking verifies a sample's trustworthiness. Poisoned data often originates from unreliable sources, and this metadata allows for its isolation and exclusion (Baracaldo et al., 2017).

Influence Functions. This principled approach estimates how each training point affects a model's predictions. Samples with disproportionately high influence can indicate harmful behaviour, allowing poisoned inputs to be detected and removed through their outsized effect on target predictions (Koh et al., 2017).

Data poisoning defences are most effective when applied in combination. Integrating data cleaning, provenance filtering, robust loss functions, and post-training influence analysis creates a more resilient training pipeline and reduces susceptibility to diverse poisoning strategies.

2.4.3 Defences Against Backdoor Attacks

Backdoor attacks embed malicious behaviours into DNNs by poisoning the training data with inputs containing specific triggers associated with target labels. Defences against such attacks aim to detect and mitigate these hidden behaviours, often by analysing model activations, identifying suspicious patterns, or sanitising the training data.

Activation Clustering. This detection strategy leverages the observation that backdoored inputs often yield distinct activation patterns in hidden layers compared to clean inputs of the same class. By extracting activations from a selected layer and applying clustering algorithms such as k-means, anomalous clusters can be isolated for further analysis. This method has demonstrated strong post-training detection capabilities (Gao; Doan, et al., 2020; W. Guo et al., 2022).

Spectral Signatures. This defence analyses the internal feature space of models, identifying outliers by applying singular value decomposition to the covariance matrix of activations. Poisoned samples often introduce dominant spectral components that deviate from clean data distributions. Once identified, these anomalies can be filtered out during training (Gao; Doan, et al., 2020; Yudong Li et al., 2023).

Anomaly Detection Algorithms. Techniques such as Isolation Forest and Local Outlier Factor offer unsupervised, model-agnostic detection of suspicious data. By evaluating the distribution

of model representations or input statistics, they help uncover poisoning traces without relying on specialised architectures (W. Guo et al., 2022; Yudong Li et al., 2023).

Pruning and Fine-Pruning. These defences focus on removing or suppressing neurons potentially responsible for malicious behaviour. Standard pruning eliminates neurons with limited contribution to clean predictions while fine-pruning retrains the network after pruning to recover accuracy and minimise backdoor effects (Gao; Doan, et al., 2020; W. Guo et al., 2022).

Model Inspection. This class of techniques analyses how models respond to crafted or controlled inputs to identify suspicious behaviours. Examples include saliency map visualisations and input reconstruction attacks. These methods can expose unintended shortcuts or overly confident responses linked to hidden triggers (Yudong Li et al., 2023).

Neural Cleanse. Neural Cleanse reverses potential triggers for each output class by optimising minimal perturbations that reliably cause misclassifications. A minimal perturbation for a specific class suggests a backdoor. The method quantifies suspicion using a statistical anomaly score (B. Wang et al., 2019).

Artificial Brain Stimulation. This defence analyses neuron sensitivity by probing the network with random synthetic inputs. Neurons linked to backdoor behaviour typically exhibit exaggerated activation patterns, which helps localise the malicious subnetwork (Y. Liu et al., 2019).

STRIP. This defence detects malicious inputs at inference time by measuring prediction entropy under strong input perturbations. When perturbed, clean inputs tend to yield variable predictions, while backdoored inputs consistently activate the target class, revealing their presence (Gao; C. Xu, et al., 2020).

These defence strategies span training-time, post-training, and inference-time solutions. Their effectiveness depends on threat model assumptions, computational overhead, and access to clean reference data. In practice, combining multiple detection and mitigation strategies often leads to stronger protection against diverse backdoor threats.

2.4.4 Defences Against Evasion Attacks

Adversarial attacks at inference time are particularly challenging due to their subtle perturbations and model transferability. Defence strategies against such attacks aim to increase model robustness through training-time adjustments, transform inputs to neutralise perturbations or provide formal guarantees on behaviour under perturbation.

Adversarial Training. This approach augments the training set with adversarial examples, forcing the model to learn more resilient decision boundaries. Common attacks such as FGSM or PGD are used during training to generate perturbed inputs. While adversarial training improves robustness significantly, it may lower clean accuracy and substantially increase computational cost (Gao; Doan, et al., 2020; W. Guo et al., 2022; Yudong Li et al., 2023).

Randomised Smoothing. Randomised smoothing constructs a robust classifier by adding Gaussian noise to inputs and aggregating predictions over multiple noisy samples. It provides provable robustness guarantees against ℓ_2-bounded perturbations and can be applied post hoc to

pre-trained models. The method is model-agnostic and compatible with standard architectures (W. Guo et al., 2022; Yudong Li et al., 2023).

Gradient Masking. These techniques obscure the gradients used by adversaries to craft perturbations, for example, through non-differentiable operations, randomness, or obfuscated loss landscapes. Although gradient masking may reduce the effectiveness of gradient-based attacks, it can often be bypassed by transfer attacks or black-box query strategies (Gao; Doan, et al., 2020; W. Guo et al., 2022).

JPEG Preprocessing. This method applies JPEG compression to inputs before inference, removing high-frequency components commonly exploited by adversarial perturbations. As a lightweight, domain-specific defence, it has shown effectiveness against many image-based attacks (Guesmi et al., 2022).

Certified Defences. Certified robustness techniques guarantee that a model's output remains unchanged within a given perturbation radius. Approaches include interval-bound propagation, convex relaxation, and Lipschitz regularisation. While promising, these methods often struggle with scalability and can limit model expressiveness (W. Guo et al., 2022; Yudong Li et al., 2023).

Input Transformation. These defences apply stochastic or deterministic changes to inputs to disrupt adversarial patterns. Common transformations include resizing, cropping, bit-depth reduction, and random masking. Stochastic Input Transformation combines several such steps to increase robustness while preserving semantic content (Guesmi et al., 2022).

Feature Squeezing. This defence reduces the input's degrees of freedom using pixel quantisation or spatial smoothing techniques. By evaluating a model's consistency on the original and squeezed input, adversarial examples can be detected or mitigated. This lightweight method integrates easily with existing pipelines (W. Xu et al., 2018).

These defences represent complementary strategies to enhance model resilience against evasion attacks. While no single technique offers complete protection, combining multiple defences tailored to specific threat models and deployment constraints can significantly reduce vulnerability.

2.4.5 Defences Against Model Extraction Attacks

Model extraction attacks aim to replicate the functionality of a target model by systematically querying it and using the responses to train a surrogate with similar behaviour. These attacks threaten intellectual property, compromise security, and may indirectly enable further privacy violations. Defence strategies against model extraction focus on monitoring query patterns, obfuscating outputs, restricting access, and embedding ownership mechanisms into models.

Query Monitoring and PRADA. PRADA is a defence that monitors the statistical properties of incoming queries to detect adversarial probing. The core assumption is that benign users issue inputs with natural variation, while adversaries tend to generate inputs that cluster near decision boundaries. PRADA computes the Euclidean distances between consecutive queries and flags deviations from expected distributions as signs of extraction behaviour (Juuti et al., 2019; Jiang et al., 2024).

Output Perturbation. This strategy seeks to reduce the precision or informativeness of the model's responses to hinder learning by an adversary. Techniques include rounding probabilities, limiting outputs to top-k predictions, or injecting calibrated noise into soft-max scores. While such perturbations may slightly reduce utility for legitimate users, they can significantly impede model replication (Juuti et al., 2019; Jiang et al., 2024).

Watermarking Models. Watermarking embeds identifiable patterns or behaviours into a model during training that can later be used to assert ownership. For instance, the model may be designed to respond in a predetermined way to a specific set of inputs (trigger set), enabling verification of misuse. This serves both as a technical deterrent and a legal safeguard against model theft (Jiang et al., 2024).

Rate Limiting and API Filtering. By capping the number of queries from a given user or slowing request rates, these mechanisms increase the cost and difficulty of extraction attacks. Some systems also apply input filtering to detect and block adversarial queries—such as those with adversarial structure or boundary-seeking patterns. These controls are particularly relevant in MLaaS contexts (Juuti et al., 2019; Jiang et al., 2024).

Model extraction defences often involve balancing usability with protection. While PRADA and output perturbation introduce minimal disruption for legitimate users, they substantially raise the barrier for adversaries. Watermarking provides an additional forensic tool to attribute model ownership and detect unauthorised use.

2.4.6 Defences Against Membership Inference Attacks

Membership inference attacks exploit the tendency of over-parameterised models to memorise training data, allowing adversaries to infer whether a specific data point was part of the model's training set. Various defence strategies have been proposed to counter this threat, aiming to reduce information leakage, control overfitting, or obfuscate model behaviour. These defences vary in complexity and effectiveness, with some offering theoretical guarantees while others rely on empirical robustness.

Regularisation and Dropout. A widely adopted strategy to defend against membership inference attacks is reducing model overfitting using standard regularisation techniques. Methods such as L2 regularisation and dropout constrain the model's capacity to memorise individual data points, thereby making membership harder to infer. Empirical evidence suggests that introducing dropout or weight decay significantly decreases the attacker's ability to distinguish between training and non-training samples (H. Hu et al., 2022; Shuvo et al., 2020).

Adversarial Regularization. This defence explicitly incorporates a simulated adversary into the training process. A regularisation term is added to the loss function to penalise the model if it becomes too distinguishable between members and non-members. The result is a minimax optimisation setup where the model is trained to perform well while simultaneously resisting inference attempts (L. Hu et al., 2023).

Differential Privacy. Differentially private training introduces noise into the gradients or outputs during training, ensuring that individual data points do not significantly influence the final

model. This approach provides formal, mathematical guarantees about data privacy and is one of the most effective defences against membership inference. However, substantial privacy budgets may come at the cost of reduced model accuracy (H. Hu et al., 2022; Shuvo et al., 2020).

Model Distillation. Knowledge distillation trains a student model to mimic the soft predictions of a larger teacher model. This process leads to smoother decision boundaries and reduced overfitting, lowering the model's sensitivity to specific training samples. As a result, the distilled student model exhibits increased resilience to membership inference attacks (L. Hu et al., 2023).

These defence strategies reflect diverse approaches, from modifying training dynamics to enforcing formal privacy guarantees. While no single method eliminates the threat, combining techniques such as dropout with DP or distillation can significantly improve resistance to membership inference in practical deployments.

2.4.7 Defences Against Model Inversion Attacks

Model inversion attacks aim to reconstruct representative input samples (e.g., images or features) from a trained model by exploiting its outputs. To counteract these privacy-invasive threats, several defence strategies have been proposed. These defences aim to reduce information leakage during inference, perturb internal representations, or constrain the query interface available to the adversary.

Access Limitation. Restricting access to the model's predictions is one of the most straightforward defences against model inversion. This can involve limiting the number of queries allowed, using coarse-grained outputs (e.g., top-k predictions or hard labels), or adding noise to confidence scores. By reducing the adversary's ability to extract fine-grained patterns from the model, such mechanisms make reconstruction significantly harder (L. Hu et al., 2023; Gong et al., 2023).

Differential Privacy. Differentially private training techniques inject calibrated noise into the training process, ensuring that the model's outputs do not depend significantly on any single input sample. As a result, the model becomes inherently more resistant to inversion attempts. This approach has been widely studied as a strong theoretical foundation for defending against membership inference and inversion attacks (L. Hu et al., 2023; Gong et al., 2023).

Adversarial Example Obfuscation. Recent studies have shown that adversarial examples, designed initially to fool classifiers, can also be used to defend against inversion attacks. Introducing targeted perturbations that degrade the attacker's ability to reconstruct meaningful inputs while preserving model accuracy effectively acts as a form of input-space camouflage (S. Zhou; Zhu, et al., 2024).

GAN-Based Obfuscation Methods. An emerging line of defence involves using GANs to transform outputs in a way that preserves task performance while obfuscating sensitive attributes. Gong et al. (2023) propose a GAN-based framework that learns to filter features exposed to the adversary, thus balancing utility and privacy. These methods provide flexible and data-adaptive defences but often require retraining or integration into the model pipeline.

These defences illustrate the evolving landscape of privacy-preserving DL. While access control and DP provide strong theoretical guarantees, obfuscation-based methods offer practical tools

for specific settings. Their effectiveness often depends on the threat model and the attacker's capabilities, reinforcing the need for a defence-in-depth approach.

The wide range of defences reviewed in this section underscores the fragmented yet active efforts to secure DL models against diverse adversarial threats. Each strategy tends to focus on a specific attack vector, and while many offer strong results under controlled assumptions, real-world deployment remains challenging. Recent research has proposed unified solutions that systematise threat identification, attack simulation, and defence evaluation to address this. These are often encapsulated in dedicated frameworks that operationalise the application of security techniques. In Section 2.5, we examine AI security frameworks embodying these goals and analyse how they integrate the previously discussed defence strategies into practical pipelines.

2.5 Existing AI Security Frameworks

The increasing prevalence of adversarial threats in DL has motivated the development of several AI security frameworks to evaluate and improve model robustness. While these tools represent important steps towards the practical deployment of secure AI systems, many exhibit scope, usability, or adaptability limitations, particularly when addressing the full spectrum of attacks and defensive strategies.

Popular open-source tools such as **Adversarial Robustness Toolbox**[7] and **CleverHans**[8] provide extensive libraries of adversarial attacks and defences. However, they often require expert knowledge to configure and interpret, lack guided workflows, and focus primarily on evasion attacks. Similarly, enterprise-focused platforms such as **IBM's AIX360**[9], **MITRE ATLAS**, and **Microsoft's Counterfit** emphasise explainability, threat cataloguing, or benchmarking, but offer limited support for automated mitigation and defence selection, particularly for non-specialists.

Several research works have proposed more integrated or specialised frameworks in response to these challenges. Sun et al. (2020) introduce a multi-stage pipeline with detection, purification, and adversarial training components, focusing on complete protection against adversarial examples. D. Li et al. (2021) propose a domain-specific defence framework for malware classification that incorporates adversarial retraining and robust feature selection. J. Guo et al. (2023) present a robustness evaluation framework encompassing white-box and black-box attacks, offering reproducible metrics across multiple threat models.

Al-Andoli et al. (2024) propose a protection-layer-based architecture to enhance DNN robustness at various stages of inference, while Arshad et al. (2023) introduce a comprehensive cyber-defence framework that incorporates monitoring, threat detection, and response for adversarial scenarios. These works collectively advance the state of AI security infrastructure but often suffer from limitations such as focusing on a narrow subset of attacks, requiring manual configuration, or lacking integration with defence recommendations and usability features.

[7] https://github.com/Trusted-AI/adversarial-robustness-toolbox
[8] https://github.com/cleverhans-lab/cleverhans
[9] https://github.com/Trusted-AI/AIX360

Furthermore, most existing frameworks do not address privacy-related threats such as membership inference, model inversion, or model extraction, nor offer risk-driven defence prioritisation or human-in-the-loop configuration. This fragmentation highlights the need for a comprehensive, modular, and user-centric framework.

To address these limitations, this dissertation introduces `Safe-DL`, a unified security framework for DL systems that integrates threat modelling, attack simulation, risk analysis, and guided defence application. Unlike previous tools, `Safe-DL` prioritises explainability, interactivity, and modularity, bridging the gap between theoretical adversarial ML research and real-world usability. The core requirements that guided its design are presented in Section 2.6.

2.6 Key Requirements for a Secure and Usable DL Framework

The extensive review of adversarial threats, defence strategies, and existing security frameworks reveals significant limitations in the current state of practice. Many tools and methods are narrowly scoped, often focusing on individual attack vectors such as evasion or poisoning, and lack integration into a cohesive, end-to-end pipeline. Furthermore, the usability of these solutions remains a persistent barrier to widespread adoption, particularly among non-expert practitioners. This fragmentation and technical inaccessibility hinder the operationalisation of adversarial robustness in real-world applications.

To address these issues, a new class of security frameworks for DL must adhere to several key requirements distilled from the gaps and challenges identified throughout this chapter. These requirements directly inform the architectural and functional principles of the `Safe-DL` framework proposed in this dissertation.

Multi-Threat Support. A comprehensive framework must address various adversarial scenarios, including training-time attacks such as data poisoning and backdoors, inference-time threats like evasion, and privacy-focused attacks involving membership inference, model inversion, and model extraction. Modular design is essential for supporting these diverse attack surfaces.

Lifecycle Coverage. Robustness should be integrated throughout the DL lifecycle—from threat modelling and attack simulation to risk analysis and defence application—ensuring that security is embedded from the outset rather than treated as a reactive patch.

Modularity and Extensibility. Given the evolving nature of adversarial machine learning, frameworks should enable easy integration of new attacks, defences, and evaluation metrics without significant refactoring. A modular architecture facilitates adaptability and fosters experimentation.

Usability for Non-Experts. Security tools must lower the barrier to entry for practitioners with limited expertise. Interactive interfaces, automated recommendations, and explainable outputs are key to enabling widespread and practical use.

Transparency and Auditability. In high-stakes or regulated environments, frameworks must provide traceable decision logs, human-readable reports, and interpretable outputs. This transparency fosters trust and supports compliance with data protection standards.

These requirements serve as the foundational principles of the `Safe-DL` framework. They encapsulate the critical capabilities necessary for a secure, practical, and adaptable DL security solution, bridging the gap between academic research and real-world deployment.

2.7 Chapter Summary

This chapter has provided a comprehensive overview of the security challenges faced by DL systems and the evolving landscape of adversarial machine learning. We began by introducing a taxonomy of adversarial threats, categorising them according to their attack vector, timing, and goals. These included data poisoning, backdoor insertion, evasion, model extraction, membership inference, and model inversion. Each threat was described in terms of its methodology, impact, and real-world relevance, illustrating the diverse ways DNNs can be compromised across the training and inference lifecycle.

We then surveyed the defence strategies proposed in the literature, structuring them by the type of threat they address. This analysis revealed a rich set of techniques, ranging from data sanitisation and robust loss functions to privacy-preserving training and adversarial input transformation, each with its strengths, limitations, and operational trade-offs. We also examined the current ecosystem of AI security frameworks, highlighting their capabilities and shortcomings, particularly regarding their usability, threat coverage, and support for automated and interpretable mitigation.

From this analysis, we derived a set of key requirements for designing a secure and usable framework for adversarial robustness. These include support for multiple threat types, full lifecycle integration, modularity, usability for non-experts, and transparency. These principles directly motivate the development of the `Safe-DL` framework, whose architecture and modules are detailed in Chapter 3, where each component is designed to operationalise these requirements within a practical and extensible security pipeline.

Chapter 3

Safe-DL Framework: Architecture and Modules

Following the threat landscape and regulatory challenges discussed in Chapter 2, this chapter presents the architecture and technical design of the proposed solution, `Safe-DL`. The goal is to provide a conceptual and functional overview of the framework, outlining how its components are structured and how they contribute to the overall adversarial security assessment pipeline.

Rather than focusing on implementation details or experimental outcomes, this chapter emphasises the guiding design principles, internal workflow, and modular interactions that support transparency, traceability, and reusability. The architecture is deliberately modular, allowing each pipeline phase, such as threat modelling, risk assessment, and defence evaluation, to be executed independently or as part of a cohesive end-to-end process.

While the full technical stack and developer-facing aspects are addressed later in the chapter, the initial sections focus on the pipeline's structure, the role of each module, and the information flow across components. The discussion includes a high-level architectural diagram and an overview of the core technologies supporting the implementation.

3.1 Architecture and Design Overview

This section presents the architecture and design principles underlying the `Safe-DL` framework. The objective is to provide a conceptual and functional understanding of how the system is structured, its components interact, and how the underlying technologies support its modularity, transparency, and traceability goals.

Rather than focusing on low-level implementation details, this section describes the pipeline's internal workflow, module responsibilities, and data flow. Particular attention is given to the shared configuration file (referred to here as the *profile file*), which is named by the user (for example, `scenario1.yaml`) and serves as the central communication channel between modules. This profile file accumulates key information throughout the pipeline, enabling reproducibility and auditability across the entire process.

The discussion begins with an overview of the system's guiding principles and logical organisation, followed by a summary of the core technologies that support the implementation. A visual representation of the complete framework architecture is also provided to illustrate the interaction flow among modules.

3.1.1 Design Principles and Workflow

The architecture of the `Safe-DL` framework is guided by four main principles: modularity, transparency, automation, and traceability. These principles were established to ensure that the system can adapt to a variety of use cases and threat models, integrate easily into real-world workflows, and provide verifiable and reproducible results.

Modularity allows each process stage to be encapsulated in an independent component with a clearly defined responsibility. This facilitates maintenance and debugging and enables users to run or extend individual parts of the framework without requiring changes elsewhere. Transparency is supported by human-readable configuration files and outputs, including YAML, Markdown, and JSON formats, making it easy to inspect, interpret, and audit each analysis step. Automation is achieved through a command-line interface (CLI) that guides the user through threat specification and executes each module based on the profile file. This ensures consistency across runs and reduces the likelihood of manual errors. Finally, traceability is ensured by a design where all critical decisions and results are progressively recorded and centralised in the profile file. This allows users to follow the entire reasoning chain and replicate any part of the pipeline if necessary.

The workflow is structured as a pipeline of six modules: threat modelling, attack simulation, risk analysis, defence application, defence evaluation, and final reporting. Each module reads specific parts of the profile file, produces its results in a dedicated subdirectory, and optionally updates the configuration with new information. Although the default execution is linear, each module can be run independently, allowing selective re-evaluation or experimentation with alternative parameters.

3.1.2 Module Organisation and Pipeline

The `Safe-DL` framework is structured as a pipeline composed of six modular components, each corresponding to a specific phase in the secure machine learning lifecycle. This modular design allows each component to focus on a distinct responsibility while enabling controlled communication and data sharing through a central YAML configuration file and a structured results directory.

Module 1 is responsible for threat modelling. An interactive CLI collects information about the system under analysis and the adversarial scenarios to be considered. The result is a structured threat profile that becomes the starting point for the pipeline.

Module 2 handles attack simulation. Based on the threat model and other profile settings, it runs one or more configured adversarial attacks. The outputs include logs, poisoned datasets, and attack-specific metadata that are saved in the results directory and referenced in the profile.

Module 3 performs risk analysis. It processes the outputs of the attacks to compute vulnerability metrics and risk scores, both globally and at the class level. These scores are written into the profile file and are later used to inform defence strategies.

Module 4 applies defences. It selects and configures appropriate mitigation mechanisms based on the computed risk levels. All defence parameters, modifications to the data or model, and any resulting metadata are stored for traceability and further evaluation.

Module 5 evaluates the effectiveness of the applied defences. It re-assesses the system using the same attack configurations and compares the results before and after mitigation. This evaluation includes both robustness metrics and associated costs.

Module 6 aggregates all previous results and produces a final report. This report is generated in Markdown format and includes summaries of the attack impact, defence outcomes, risk scores, and relevant visualisations.

All modules read from and write to the profile file, storing their outputs in dedicated `results/` subdirectories within each module's folder. This structure ensures that outputs remain encapsulated, traceable, and easily managed across the pipeline. Although the pipeline is typically executed linearly, the independence between modules allows any of them to be run separately, supporting incremental development, targeted re-evaluation, or experimentation with alternative configurations without requiring full re-execution.

3.1.3 Core Technologies and Stack

The `Safe-DL` framework is implemented in Python 3.10 and builds upon a set of widely adopted libraries in the machine learning and scientific computing ecosystem. This technological base was selected to ensure modularity, extensibility, and ease of development while enabling reproducible and auditable security assessments.

PyTorch serves as the backbone of the DL components, powering model training, evaluation, and gradient-based adversarial methods. **Scikit-learn** supports metric computation, data preprocessing, and auxiliary model utilities. For configuration handling, the system relies on **PyYAML**, which allows reading and updating the profile file in a structured and human-readable format.

The CLI is developed with **questionary**, providing an interactive environment for threat modelling that simplifies the configuration process. Visual artefacts are produced using **matplotlib**, while structured outputs such as logs and metrics are saved in **JSON** format to enable automated analysis and inspection. Additionally, **tabulate** generates well-formatted Markdown tables, making the final reports easier to read and integrate into documentation workflows.

All components are organised in a modular fashion, and the codebase is structured to facilitate the addition of new datasets, models, attacks, or defences without modifying core logic. The choice of stable and well-supported libraries ensures that the system remains compatible with current tools and practices in the field.

3.1.4 System Architecture Diagram

To support the conceptual overview provided in the previous sections, this subsection introduces a visual representation of the Safe-DL framework's architecture. The diagram in Figure 3.1 illustrates the sequence of modules, their responsibilities, and how they communicate via shared files and structured outputs.

Each module is an independent block interacting with two key artefacts: the central profile configuration file on the left and the module-specific results directory on the right. The arrows indicate the direction of data flow, where blue corresponds to reading from the profile, red corresponds to writing into it, orange corresponds to generating output artefacts, and purple corresponds to dependencies on previous results. This precise mapping makes the internal workflow traceable and facilitates the reproducibility of any given analysis.

This system-level perspective helps clarify the structure and responsibilities of the pipeline, preparing the ground for the more detailed descriptions of each module that follow.

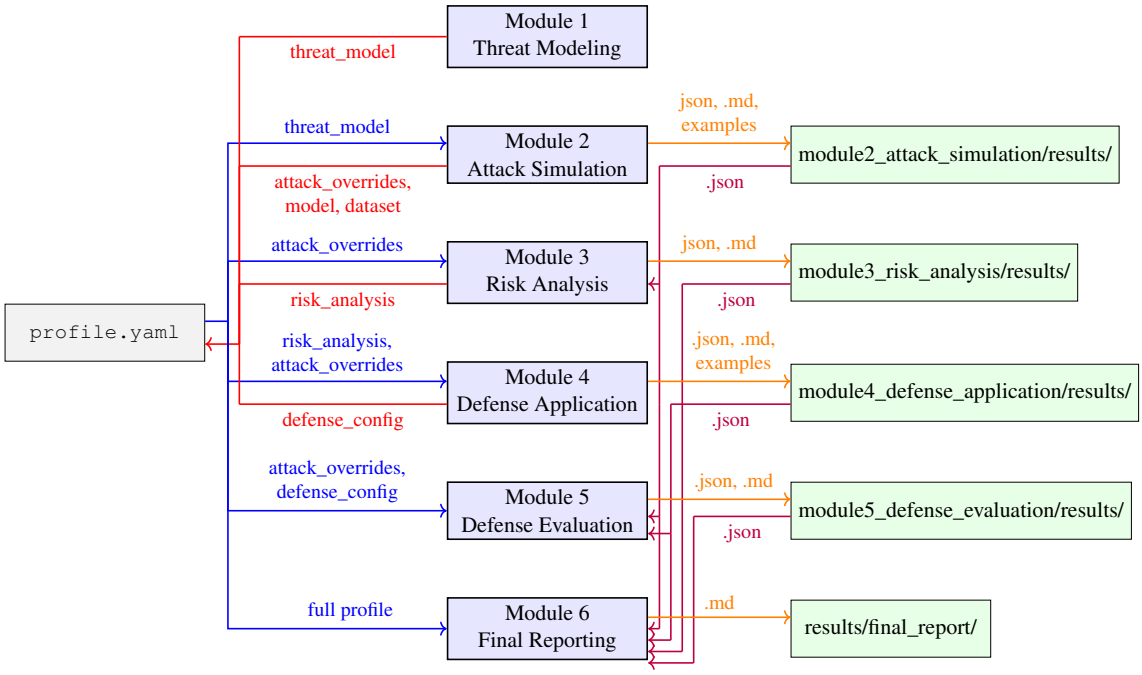


Figure 3.1: Modular architecture of the Safe-DL framework, showing how each module interacts with the profile configuration and its dedicated results output. Arrows denote read, write, and generation flows across the pipeline.

With the overall architecture, design principles, and technological foundations now established, the next sections provide a detailed breakdown of each module in the Safe-DL pipeline. We

begin in Section 3.2 with the threat modelling phase, which defines the adversarial context and initiates the security assessment process.

3.2 Module 1 — Threat Modeling

The first module in the `Safe-DL` pipeline is responsible for eliciting and formalising the adversarial context in which the system operates. It collects user input through an interactive command-line questionnaire and generates a structured YAML profile that captures security-relevant properties of the model, data, and deployment environment. This profile becomes the foundational input for the remainder of the pipeline, influencing downstream behaviour and ensuring that assumptions are consistently propagated.

The module has no dependencies on previous components, as it is designed to be the natural entry point of the system. Its output is consumed directly by Module 2, which uses the inferred threat categories to select appropriate attack simulations. Module 6 also references this profile during the final report generation to ensure traceability between the defined threat model and the resulting security assessment.

To maximise usability, the profile is generated through an interactive CLI. The user is prompted to answer a sequence of contextualised questions, and the system uses the responses to infer a preliminary list of relevant adversarial threats. This module is executed via the `run_module1.py` script, which launches the questionnaire and saves the resulting data in the `threat_model` section of the central profile file. This file is updated and reused throughout the pipeline. The generated profile includes the following fields:

Model access. Specifies the level of access an adversary is assumed to have over the model. Three options are supported: `white-box` grants complete visibility into the model's architecture, parameters, and training data; `gray-box` allows partial insight, such as knowledge of the architecture without access to weights; and `black-box` restricts the attacker to observing only input-output behaviour. This setting is critical for determining which attacks are feasible in later stages.

Attack goal. Defines the intended outcome of an adversarial attempt. A `targeted` attack aims to force the model into a specific misclassification (e.g., always predicting "cat"), whereas an `untargeted` attack seeks to degrade performance by causing any incorrect output. This distinction informs the choice of attack methods and the evaluation metrics used.

Deployment scenario. Indicates where the model will be deployed and operated. Available options include `cloud` (centralised and remotely accessible), `edge` (deployed near the data source, e.g., on local gateways), `mobile` (smartphones or tablets), `api_public` (exposed through open APIs), and `on_device` (entirely local inference with no remote access). These settings influence the attack surface and threat exposure.

Data sensitivity. Captures the confidentiality level of the training data. `High` sensitivity includes personal or medical records; `medium` includes internally relevant but non-confidential

datasets; and `low` refers to public, anonymised, or synthetic data. This parameter is particularly relevant for privacy-preserving defences.

Training data source. Describes the origin and trust level of the training dataset. Accepted values are `internal_clean` (fully curated and verified data), `external_public` (public datasets with uncertain provenance), `user_generated` (data collected from users or crowdsourcing), and `mixed` (combinations of trusted and untrusted sources). This field directly affects the likelihood of data poisoning.

Model type. Indicates the family of architectures used. Choices include `cnn` (typically for vision), `transformer` (for sequential or multi-modal tasks), `mlp` (multi-layer perceptrons for structured data), and `other` for custom or hybrid models. Depending on the model class, some attacks and defences may be more or less effective.

Interface exposed. Describes the primary method through which the model is accessed during inference. Options include `api` (HTTP or RESTful endpoints), `local_app` (embedded in desktop or mobile apps), `sdk` (software libraries with exposed APIs), and `none` (fully offline use). Interfaces with remote access may enable threats such as model stealing or membership inference.

Threat categories. A list of high-level attack types inferred from the answers provided above. This field may include `data_poisoning`, `backdoor_attacks`, `evasion_attacks`, `model_stealing`, `membership_inference`, and `model_inversion`. These categories guide the attack simulations executed in Module 2. [1]

The output of this module is a structured YAML file that captures the complete threat profile and acts as a foundational input for the subsequent stages of the `Safe-DL` pipeline. Formalising security assumptions in a reproducible format ensures consistency, traceability, and alignment across the framework. In particular, the inferred threat categories are directly consumed by Module 2 to configure and trigger relevant attack simulations, as described in Section 3.3.

3.3 Module 2 — Attack Simulation

The second module in the `Safe-DL` pipeline is responsible for simulating adversarial attacks against the target model. Its primary purpose is to reproduce realistic threat scenarios that reflect the assumptions defined in the preceding threat model. This enables the subsequent evaluation of a model's robustness and the effectiveness of defence mechanisms.

This module directly depends on the `threat_categories` field generated by Module 1, which determines the types of attacks to be executed. Its outputs form a critical foundation for the remainder of the pipeline. Module 3 uses the generated results to quantify the model's vulnerability and risk exposure. Module 4 reuses the attack configurations to evaluate defensive strategies under the same conditions. Module 5 analyses the effectiveness of those defences by comparing pre-and post-mitigation metrics, and Module 6 incorporates all outcomes into the final

[1] While privacy-related categories such as `model_stealing`, `membership_inference`, and `model_inversion` are supported at the threat modelling level, they are not yet implemented in later modules of the current version of `Safe-DL`.

consolidated security report. As such, this module acts as a central operational hub linking threat assumptions to empirical evaluation.

The attack simulation process is split into two distinct phases to ensure modularity and extensibility. First, a configuration phase collects user-defined parameters regarding datasets, models, and attack-specific settings. These are stored in the `attack_overrides` section of the central profile file. Second, an execution phase runs the selected attacks independently, logging outputs in a structured format for later consumption.

Three categories of adversarial attacks are currently supported: **data poisoning**, **backdoor attacks**, and **evasion attacks**. A dedicated submodule handles each category, and all share a unified input/output interface that simplifies orchestration, traceability, and future extension. The rest of this section presents these phases in detail, beginning with the setup process.

3.3.1 Setup Phase

Before any attacks can be executed, Module 2 must be configured through an interactive setup process. This step defines the dataset, model architecture, and parameters for each attack category identified in the threat profile from Module 1. All user choices are stored in the shared `<profile_name>.yaml` file, ensuring consistency and reproducibility throughout the framework.

Configuration process: The user is guided through a sequence of terminal prompts by the `setup_module2.py` script. These include selecting a dataset (either built-in or custom), choosing a model architecture, and setting hyperparameters for each attack class. When possible, default values are automatically inferred from the dataset and threat model but may be overridden.

Supported built-in datasets: `Safe-DL`includes native support for several standard datasets: `mnist`, `fashionmnist`, `kmnist`, `cifar10`, `cifar100`, `svhn`, and `emnist` (balanced). These datasets were selected due to their widespread use in the literature on adversarial machine learning, their availability in open-source platforms, and their diversity in complexity. For instance, `mnist`-like datasets offer low-dimensional grayscale images ideal for quick prototyping and controlled experiments, while `cifar10` and `svhn` present higher visual complexity and enable more realistic robustness assessments. This diversity supports evaluation across a range of difficulty levels, contributing to the generalisability of experimental results. All built-in datasets are automatically downloaded, normalised using standard statistics, and split into training, validation, and test sets. Optional data augmentation is applied to the training set when enabled.

Custom datasets: Users can provide their own datasets by implementing a `get_dataset()` function in `user_dataset.py`. The function must return the train, validation, and test splits, the number of classes, and class labels. This extensibility mechanism ensures that `Safe-DL`is not limited to academic datasets, allowing seamless integration of real-world or proprietary data. By following a simple interface, practitioners can evaluate adversarial robustness in application-specific contexts without modifying the core framework.

Supported built-in models: The framework includes several built-in architectures: `cnn`, `mlp`, `resnet18`, `resnet50`, and `vit`. These models were selected to represent a gradient of architectural complexity, ranging from shallow networks suited to small-scale datasets, to deep residual and transformer-based architectures commonly used in production systems. This variety allows for benchmarking the effectiveness and generalisation of adversarial attacks and defences across different model types. All models are initialised from scratch and adapted to the dataset's shape and number of classes. When selected, default parameters such as hidden sizes or filter counts are automatically set unless explicitly overridden.

Custom models: Custom architectures can be loaded by defining a `get_model()` function in `user_model.py`, which returns a valid `torch.nn.Module`. Optionally, a `features()` method can also be defined to support feature-space attacks like clean-label poisoning. This design decision ensures that researchers and practitioners can easily plug in novel or domain-specific models, including those requiring internal access to feature representations. The ability to define custom models without modifying internal logic is critical for supporting a wide range of research and deployment scenarios.

Profile integration: All selected configurations are saved into the central profile file. In particular, the `dataset`, `model`, and `attack_overrides` sections are populated during this phase. These entries are then used as authoritative inputs by the corresponding attack submodules.

Reusability: Once generated, a profile can be reused across different runs, manually edited, or cloned for new experiments. This allows users to iterate on model and attack configurations without repeating the setup process, ensuring consistent and comparable results.

This setup phase acts as a bridge between high-level threat assumptions and low-level attack execution. It enables `Safe-DL` to convert abstract risk factors into concrete experimental configurations. The resulting profile file becomes a central coordination hub used by every submodule in Module 2 and subsequent modules such as evaluation, defence, and reporting. Once the configuration is complete and saved to the `<profile_name>.yaml` file, the full suite of attacks can be launched by running the `run_module2.py` script. This command automatically selects and executes the relevant attack submodules based on the `attack_overrides` field, storing results in structured directories for subsequent analysis.

3.3.2 Data Poisoning Attacks

Data poisoning attacks aim to compromise the integrity of a machine learning model by injecting malicious samples into the training dataset. These attacks are designed to influence the model's behaviour at inference time while remaining undetected during training or validation. In `Safe-DL`, two classes of poisoning attacks are implemented: `label_flipping` and `clean_label`. Both are automatically configured based on the threat profile and manually refined during the setup phase. Configuration details are stored in the `attack_overrides` section of the profile file.

Label flipping attacks alter the labels of selected training samples while leaving their pixel data untouched, simulating threats to the data annotation process without requiring access to the model internals. The `strategy` parameter defines how the labels are flipped: `fully_random`

replaces each label with a random incorrect class; `many_to_one` redirects multiple source classes to a single, common target class; and `one_to_one` maps a specific source class to a specific target class. Additional parameters include `flip_rate`, which sets the proportion of the dataset to modify, and optional class indices such as `source_class` and `target_class` to control the mapping behaviour in more targeted scenarios.

Clean-label poisoning attacks perturb the pixel content of selected training samples while preserving their original labels, making them harder to detect and more suitable for stealthy adversaries. The `perturbation_method` field specifies the transformation applied to poisoned samples: `overlay` superimposes a visual pattern; `feature_collision` aligns the sample's internal representation with that of the target class; and `noise` injects subtle Gaussian perturbations. Key parameters include `fraction_poison` to define the portion of data to modify, `target_class` to indicate the desired misclassification output, `epsilon` and `max_iterations` to control the strength and duration of the perturbation process, and `source_selection` to determine which base samples are chosen for poisoning.

After execution, each attack generates a structured `metrics.json` file and a corresponding human-readable `report.md`. These outputs capture the core evaluation results and summarise the effectiveness of the poisoning process. Key metrics include `accuracy_after_attack`, which reports the model's final accuracy on a clean test set; `per_class_accuracy`, which breaks down performance by class to highlight uneven degradation; and either `flip_rate` (for label flipping) or `fraction_poison` (for clean-label attacks), which indicate the proportion of poisoned samples in the training set. Relevant class indices such as `target_class` and `source_class` are also recorded to describe the intended misclassification behaviour. A selection of illustrative example modifications, accessible via the `example_flips` or `example_poisoned_samples` fields, illustrates how training data was altered in practice.

All results are stored in `module2_attack_simulation/results/data_poisoning/`, with separate subfolders for each attack type: `label_flipping/` and `clean_label/`. Each subfolder contains a structured metrics file, a Markdown summary report, and a folder with visual examples of poisoned or relabelled images.

Figure 3.2 shows one such visual output, in which a CIFAR-10 image labelled initially as "cat" has been flipped to "frog".

3.3.3 Backdoor Attacks

Backdoor attacks compromise a model by embedding a hidden trigger into the training data that causes the model to misclassify inputs into a specific target class when present at inference time. These attacks are designed to preserve high performance on clean data while introducing targeted vulnerabilities. `Safe-DL` supports two types of backdoor attacks: one using a static visual patch and another based on a learned optimised trigger.

Static patch backdoor attacks inject a fixed visual trigger into a portion of the training set, such as a white square, checkerboard, or random noise. This patch is blended into the image with a configurable transparency level using the `blend_alpha` parameter. The model is trained to

Figure 3.2: Example of label flipping: original label `cat` flipped to `frog`.

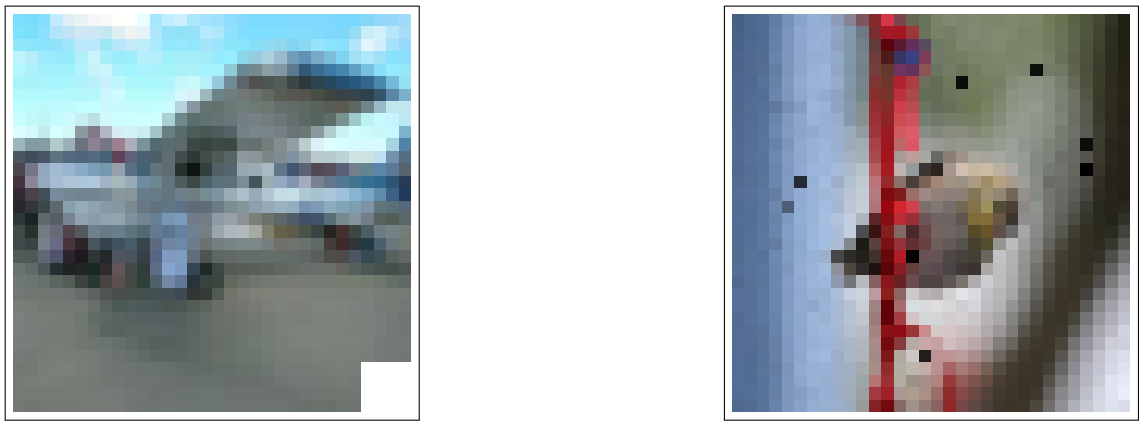
associate the presence of this patch with a specific `target_class`, resulting in misclassification when the trigger is present at test time. The `label_mode` controls whether the poisoned images are relabelled (`corrupted`) or retain their original label (`clean`), balancing stealthiness and attack success. Additional parameters include `patch_type`, which determines the appearance of the trigger; `patch_position`, which specifies its location in the image; `patch_size_ratio`, defining its relative size; and `poison_fraction`, which sets the proportion of the training dataset that is modified.

Learned trigger backdoor attacks use optimisation to generate a custom perturbation and mask that act as the trigger. These components are trained to maximise misclassification toward a specific `target_class` while remaining as imperceptible as possible. The attack always operates in `corrupted` labelling mode. Key parameters include `patch_size_ratio`, which limits the trigger size; `poison_fraction`, controlling how many training samples are altered; and optimisation hyperparameters like `learning_rate` and `epochs`. Regularisation terms `lambda_mask` and `lambda_tv` are also available to enforce sparsity and smoothness in the generated patch.

After execution, each attack produces a structured `metrics.json` file and a human-readable `report.md`. These files summarise both configuration settings and evaluation results. Key metrics include `accuracy_clean_testset`, which measures model performance on unaltered data; `attack_success_rate` (ASR), which quantifies how often poisoned inputs lead to misclassification; `per_class_clean`, showing class-wise clean accuracy; and `per_class_attack_success_rate`, which breaks down ASR per source class. A selection of example modifications, accessible via the `example_poisoned_samples` field, illustrates how the triggers affect inputs.

All outputs are saved under the `module2_attack_simulation/results/backdoor/` directory. Subfolders include `static_patch/` and `learned_trigger/`, each containing the corresponding metrics file, a Markdown summary report, and a folder with visual examples of poisoned inputs.

Figure 3.3 shows one poisoned CIFAR-10 image from each variant. On the left, a white square trigger redirects a sample labelled initially as "airplane" to "automobile". On the right, a learned perturbation applied to a bird image causes it to be misclassified as "horse".



(a) Static Patch Backdoor (b) Learned Trigger Backdoor

Figure 3.3: Examples of poisoned samples from the CIFAR-10 dataset generated by backdoor attacks in Module 2.

3.3.4 Evasion Attacks

Evasion attacks attempt to mislead a trained model by applying small, carefully designed perturbations to test-time inputs. Unlike poisoning or backdoor threats, evasion occurs purely at inference time and does not alter the training data. `Safe-DL` implements a range of evasion techniques grouped into white-box, black-box, and transfer-based categories.

White-box attacks assume full access to the model and its gradients. The `fgsm` attack applies a single-step perturbation bounded by `epsilon`, which defines the maximum change applied to each pixel in the input image. The goal is to increase the model's loss and induce misclassification in a single gradient step. The `pgd` variant improves upon FGSM by applying multiple iterations of smaller perturbations. Its behaviour is controlled by `epsilon`, the total perturbation budget; `alpha`, the step size in each iteration; `num_iter`, the number of iterations to perform; and `random_start`, which determines whether the attack should begin from a randomly perturbed input within the epsilon ball. The `cw` attack formulates evasion as an optimisation problem that minimises the L2 distance between clean and adversarial inputs. Its configuration includes `confidence`, which sets the required confidence margin in favour of the adversarial class; `binary_search_steps`, the number of steps used to tune regularisation strength; `learning_rate`, which governs the speed of optimisation; `initial_const`, the starting weight of the attack loss term; and `max_iterations`, the total number of update steps. Finally, `deepfool` iteratively moves inputs across decision boundaries with minimal perturbation, tuned using `overshoot`, which slightly amplifies the final step to guarantee misclassification, and `max_iter`, which limits the number of refinement steps.

Black-box attacks do not require access to model internals. The `nes` attack approximates gradients using stochastic sampling. It is governed by `epsilon`, the maximum allowed perturbation; `sigma`, which defines the standard deviation of noise used in gradient estimation; `learning_rate`, the step size in parameter updates; `num_queries`, the number of model queries used in total; and `batch_size`, which sets how many perturbed samples are averaged per step. Similarly, the `spsa` method estimates gradients via two-sided finite differences. Its key parameters are `epsilon`, the perturbation bound; `delta`, the magnitude of small perturbations used in estimation; `learning_rate`, which controls the update strength; `num_steps`, the total number of iterations; and `batch_size`, which determines the number of samples used to average each step.

Transfer-based attacks attempt to generate adversarial examples using a substitute model, hoping they will generalise to the target model. These attacks are configured using `substitute_model`, which defines the architecture used for surrogate training, and `attack_method`, which specifies the evasion technique to apply (e.g., `fgsm`, `pgd`) against the substitute.

After execution, each attack produces a structured `metrics.json` file and a corresponding human-readable `report.md`. These outputs summarise configuration parameters and evaluation results. Key metrics include `accuracy_clean_testset`, which reports performance on unperturbed test samples, and `accuracy_adversarial_testset`, reflecting performance on perturbed inputs. The framework also includes `per_class_clean` and `per_class_adversarial` for class-wise analysis and provides `example_adversarial_samples` for visualising how inputs were modified.

All results are stored under the `module2_attack_simulation/results/evasion/<attack_name>/` directory. Each subfolder contains the `<attack_name>_metrics.json` file, a `<attack_name>_report.md`, and an `examples/` directory with visual artifacts.

An illustrative example of this transformation is presented in Figure 3.4, where the model's prediction changes from "frog" to "deer" after the PGD attack.

(a) Original Image (b) Adversarial Image (PGD)

Figure 3.4: Example of adversarial perturbation on a CIFAR-10 image using the PGD attack.

Taken together, data poisoning, backdoors, and evasion attacks offer a comprehensive assessment of the model's vulnerability surface under diverse adversarial conditions. All attack results are stored in a structured format and reused in downstream stages of the `Safe-DL` pipeline. In particular, the metrics and artefacts produced here are directly consumed by Module 3, where the observed performance degradations are analysed to quantify the severity and scope of each identified threat. We now turn to this analysis phase in Section 3.4.

3.4 Module 3 — Risk Analysis

After simulating adversarial attacks in Module 2, the next step in the `Safe-DL` pipeline is to assess the potential risk posed by each identified threat. Module 3 performs this task by analysing the outputs of all executed attacks, calculating relevant metrics, and assigning a risk score to each one. It directly depends on the attack reports, metrics, and configuration parameters produced by Module 2, and its results serve as a foundation for both Module 4, which selects and applies defences based on the computed risks, and Module 6, which integrates all findings into the final system-wide security report.

The primary goal of this module is to bridge the gap between raw technical outputs and actionable security insights. While Module 2 produces empirical evidence of vulnerabilities, these results alone do not indicate which threats are most critical or require urgent mitigation. Module 3 transforms these isolated outputs into structured, comparable risk assessments that inform downstream planning.

To perform its analysis, `Safe-DL` collects the structured JSON outputs generated by each attack in Module 2 and the corresponding attack configuration parameters saved in the `attack_overrides` section of the central profile file. These dual inputs allow the framework to account for the observed impact of the attack and the conditions under which it was executed. Specifically, it extracts values such as clean accuracy, attack success rate, class-wise performance breakdowns, and visibility metrics where available.

The framework quantifies risk along three principal axes: *severity*, *probability*, and *visibility*. Severity captures the magnitude of degradation in model performance. For evasion attacks, this is the drop in accuracy between clean and adversarial inputs. For backdoors, it is measured using the ASR. For poisoning attacks, it reflects the drop in clean accuracy following training on corrupted data. All severity values are normalised to the $[0, 1]$ range.

Probability estimates how likely the attack is to succeed in realistic conditions. It is inferred from output metrics and threat context, such as high ASR or low adversarial accuracy. At present, `Safe-DL` uses rule-based heuristics with configurable thresholds to compute this dimension, but the framework is designed to accommodate more sophisticated estimators in the future.

Visibility reflects how noticeable the attack is to a human observer or automated detection system. A precomputed score is extracted from the attack's output file (e.g., based on L_2 norm or patch conspicuity) if available. Otherwise, a default visibility level is assigned based on the type

of attack: high for conspicuous triggers like static patches, medium for evasion perturbations, and low for stealthy methods like clean-label poisoning.

These three axes are then combined into a final risk score using Equation 3.1.

$$\text{Risk Score} = \text{Severity} \times \text{Probability} \times (1 + (1 - \text{Visibility})) \tag{3.1}$$

This formulation assigns higher scores to attacks that are both effective and stealthy. For example, an attack with severity $= 1.0$, probability $= 0.9$, and visibility $= 0.2$ would yield a final score of $1.0 \times 0.9 \times (1 + 0.8) = 1.62$.

Once computed, these scores are saved in a machine-readable `risk_analysis.json` file and a human-readable Markdown report `risk_report.md`. The former includes all computed metrics and metadata, while the latter summarises the results in tables, qualitative matrices, and ranked lists. In addition, the profile file is updated to include a new `risk_analysis` section containing each attack's score and a list of recommended defences. This section is structured into two fields: `summary`, which maps each attack to its severity, probability, visibility, and final score, and `recommendations`, which lists the defence strategies selected for each attack.

The Markdown report includes metadata such as the dataset, model architecture, and timestamp, as well as a summary table of all attacks with direct links to their reports. A qualitative risk matrix categorises threats by severity and probability, and a risk ranking lists them in descending order of computed score. For each attack, tailored recommendations are generated automatically using rule-based logic. For instance, high-ASR backdoors may trigger suggestions such as activation clustering or model inspection; strong PGD evasion attacks might lead to adversarial training or randomised smoothing; and impactful clean-label poisoning may call for influence function analysis or data provenance tracking. Redundant recommendations are deduplicated before being rendered.

The analysis process is triggered by executing the `run_module3.py` script, which automatically collects the necessary outputs, applies the scoring logic, and generates the structured risk reports. These results serve as a critical foundation for the defence planning phase introduced in Section 3.5, where threats are addressed based on their computed severity, likelihood, and stealthiness. Module 3 acts as the operational bridge between technical vulnerability assessment and actionable mitigation strategy within the `Safe-DL` pipeline.

3.5 Module 4 — Defence Application

After identifying and prioritising adversarial threats in Module 3, the fourth stage of the `Safe-DL` pipeline focuses on applying targeted defences to mitigate those risks. Rather than relying on manual selection, the system leverages the `risk_analysis.recommendations` section of the profile file to suggest which defences should be deployed for each attack automatically.

This module builds on the structured risk assessments produced in Module 3 and uses the attack configurations from Module 2 to re-simulate the relevant adversarial scenarios under defence

conditions. This ensures that each mitigation strategy is tested in a realistic and reproducible threat environment. The resulting outputs serve as inputs for Module 5, which systematically evaluates the effectiveness of each defence, and for Module 6, which consolidates the whole security posture into a final report. As such, Module 4 acts as a critical bridge between diagnosis and validation in the `Safe-DL` workflow.

Defences in `Safe-DL` are organised into three main categories, mirroring the adversarial taxonomy established in previous modules. Data poisoning defences aim to remove or neutralise corrupted training samples. Backdoor defences attempt to detect and suppress hidden triggers embedded during training. Evasion defences strengthen the model against test-time adversarial perturbations through robust training or inference-time preprocessing.

The rest of this section explains how defences are configured and applied, beginning with the setup phase.

3.5.1 Setup Phase

Before applying any mitigation strategies, Module 4 requires user input to determine which defences should be activated. While the `risk_analysis` section of the central profile file provides a list of recommended defences tailored to the specific attack landscape identified in Module 3, these suggestions are not enforced. Instead, the user is guided through an interactive setup process that offers flexibility and transparency.

Interactive configuration. The setup is performed by executing the `setup_module4.py` script, which parses the list of detected attack types and presents a corresponding list of suggested and available defences for each one. For every threat category (data poisoning, backdoor, and evasion), the user can either accept the recommended configuration or customise the selection by enabling or disabling specific techniques.

Parameter specification. The system configures their associated hyperparameters once the defences have been selected. A predefined set of configuration prompts is displayed for each defence, allowing the user to tailor parameters such as thresholds, perturbation bounds, pruning ratios, or algorithmic modes. Default values are provided for convenience but may be overridden to suit the user's security and performance goals.

Profile integration. All configuration choices are recorded in the `defense_config` section of the profile file. This section stores the list of selected defences to apply for each relevant attack type, the hyperparameters associated with each defence, and a mapping from each defence to its corresponding sub-attack or threat category.

This design ensures full traceability and reproducibility across pipeline executions. It also enables precise mapping between observed risks and applied mitigations, essential for later evaluation and reporting phases. The resulting configuration is the authoritative input for all automated defence routines launched in the next phase.

3.5.2 Data Poisoning Defences

To mitigate the impact of data poisoning attacks simulated in Module 2, the `Safe-DL` framework provides six mitigation techniques. Each defence is configured during the setup phase based on the type of attack identified in the threat profile, and its execution is guided by the recommendations produced in Module 3. The outputs of each defence are stored in a structured format for downstream consumption in the evaluation and reporting phases.

The framework applies different defences depending on the subcategory of data poisoning threat. For label flipping attacks, the selected defences include Data Cleaning, Per-Class Monitoring, Robust Loss, and Differential Privacy Training. For clean-label poisoning attacks, the recommended strategies shift towards Provenance Tracking, Influence Functions, Robust Loss, and again Differential Privacy Training.

Data Cleaning. This defence aims to eliminate anomalous training samples that behave differently from the rest of the dataset. Two strategies are supported: loss-based filtering, which removes samples with abnormally high training loss, and feature-based outlier detection, which identifies data points whose representations deviate significantly from class-wise centroids. The user configures the `method` field to select between the two strategies and defines the `threshold`, which sets the percentile cutoff for removal (e.g., 0.9 removes the top 10% most anomalous samples).

Per-Class Monitoring. This technique detects poisoned classes by inspecting the intra-class loss distribution. Poisoned classes tend to exhibit greater variance or skew in their loss curves due to the presence of corrupted labels or data points. The defence uses a single parameter, `std_threshold`, to define the standard deviation level above which a class is considered suspicious. This defence does not remove data but flags anomalies for further analysis or filtering.

Robust Loss. Instead of removing data points, this method improves resilience to label noise by replacing the default loss function with a more tolerant alternative during training. The available options are: Generalized Cross Entropy, which down-weights uncertain predictions; Symmetric Cross Entropy, which balances clean and noisy contributions; and Label Smoothing, which softens target distributions. The user specifies the `type` of the loss function, and, depending on the choice, additional internal hyperparameters may be adjusted automatically.

Differential Privacy Training. Developed initially for privacy protection, this technique reduces the influence of individual samples by applying Differentially Private SGD. It achieves robustness by clipping gradients and injecting calibrated noise during optimisation. Three parameters are required: `epsilon`, which controls the privacy budget and also the noise scale; `delta`, representing the allowable probability of privacy failure; and `clip_norm`, which sets the maximum L2 norm of gradients before noise is added.

Provenance Tracking. This defence attempts to identify suspicious subsets of the training data by analysing sample origin or grouping characteristics. It assumes that poisoned samples often come from different sources or exhibit consistent patterns. The defence supports two modes of operation: `sample`-level tracking, which inspects each point individually, and `batch`-level

tracking, which aggregates data over mini-batches. The user selects the desired granularity via the `granularity` field.

Influence Functions. This method estimates how much each training point contributes to specific model predictions. It is particularly effective for identifying clean-label poisoning, where a few crafted samples disproportionately affect the decision boundary. Two estimation modes are available: `grad_influence`, which uses fast gradient approximations, and `hessian_inverse`, which offers more accurate but computationally intensive influence estimates. The user specifies the `method` and the `sample_size`, which controls how many training samples are evaluated for influence.

After execution, each defence stores its results in a well-structured output directory at `module4_defense_application/results/data_poisoning/<defense_name>/`. This directory includes a `<defense_name>_results.json` file containing post-mitigation metrics such as `accuracy_clean` and `per_class_accuracy_clean`. If applicable, the file also includes artefacts like removed or flagged samples affected by the mitigation. In addition, a human-readable summary is written to `report.md`, detailing the configuration parameters and the overall effectiveness of the defence.

3.5.3 Backdoor Defences

To mitigate backdoor threats, where hidden triggers embedded in training data cause targeted misclassification at inference time, the `Safe-DL` framework provides six defence techniques. These methods are applied to static patch and learned trigger variants and configured based on the threat profile generated earlier in the pipeline.

Activation Clustering. This technique clusters internal model activations extracted from training samples, assuming poisoned and clean inputs activate the model differently. It typically uses unsupervised clustering (e.g., K-Means) with two clusters to distinguish clean from poisoned examples. The main parameter is the `num_clusters`, which controls the number of groups to form during analysis. A value of 2 is most common, corresponding to the clean versus poisoned dichotomy.

Spectral Signatures. This defence leverages singular value decomposition on hidden layer activations to detect anomalous patterns introduced by poisoned data. Poisoned samples often exhibit strong projections onto the top singular vectors, which can be used to separate them from regular inputs. The key configuration is the `threshold` value, which sets the cutoff above which samples are flagged as suspicious based on their spectral scores. A default value of 0.9 is commonly used.

Anomaly Detection. This method applies classical outlier detection algorithms to the model's internal representations. It relies on the assumption that poisoned samples deviate significantly from the normal distribution of clean data. Users select the algorithm to apply using the `type` parameter, choosing between options such as `isolation_forest` and `lof` (Local Outlier Factor), each with different strategies for identifying anomalies in high-dimensional space.

Pruning. This defence suppresses the impact of poisoned behaviour by removing neurons suspected of encoding the backdoor trigger. It can be applied globally across all layers or limited to the final layer. The `pruning_ratio` parameter defines the fraction of neurons to remove, while `scope` controls whether the pruning targets all layers (`all_layers`) or only the output layer (`last_layer_only`). The goal is to weaken the backdoor activation pathway without degrading overall performance.

Fine Pruning. An extension of the pruning approach, fine pruning targets neurons with low activation levels on clean data. These neurons are assumed to be dormant in regular operation but may become active in the presence of a trigger. The key parameter is again the `pruning_ratio`, which determines the fraction of the least active neurons to eliminate. This more targeted pruning aims to minimise collateral damage to model accuracy.

Model Inspection. This technique inspects the model's learned parameters, namely weights and biases, to detect statistically anomalous values that may indicate backdoor mechanisms. The user provides a list of layers to inspect via the `layers` parameter (e.g., `conv.0, fc.1`), which is expanded internally to include both weight and bias tensors. Suspicious patterns in these parameters may suggest tampering or unusual training dynamics.

Each defence stores its results in an output directory at `module4_defense_application/results/backdoor/<defense_name>/`. This includes a `<defense_name>_results.json` file containing post-mitigation metrics such as `accuracy_clean` and `asr_after_defense`, along with per-class variants like `per_class_accuracy_clean` and `per_original_class_asr`. The directory includes artefacts such as filtered samples or inspection logs, if applicable. A Markdown summary in `report.md` complements the JSON data by describing the defence configuration and summarising its impact.

3.5.4 Evasion Defences

To mitigate adversarial examples that target the model at inference time, the `Safe-DL` framework provides four evasion defences. These methods are tailored to counter different classes of attacks, such as gradient-based white-box evasion and black-box score-based attacks. Each defence can be configured via interactive prompts during setup and is selected based on the threat landscape identified in previous modules.

Adversarial Training. This method retrains the model using a mixture of clean and adversarial examples, improving robustness against the specific perturbation patterns used in common attacks such as FGSM and PGD. Adversarial samples are generated on the fly during training. The user selects the base attack using the `attack_type` parameter, which can be set to `fgsm` or `pgd` and controls the strength of perturbations via the `epsilon` parameter. A higher epsilon typically results in stronger robustness but may reduce clean accuracy.

Randomized Smoothing. This technique adds Gaussian noise to inputs at inference time and aggregates the predictions over multiple noisy variants to improve the model's robustness. It is based on the principle of certified robustness, offering probabilistic guarantees under certain assumptions. The key parameters are `sigma`, which defines the standard deviation of the noise

distribution, and `num_samples`, which sets the number of noisy versions generated per input. Larger values of `num_samples` generally increase robustness but incur higher inference cost.

Gradient Masking. This defence reduces the effectiveness of white-box attacks by distorting the model's gradients, making it harder for adversaries to compute effective perturbations. This is achieved by injecting noise or applying transformations to the gradient flow. The user configures this method using the `strength` parameter, which determines the intensity of gradient disruption. A higher value increases masking but may affect model performance.

JPEG Preprocessing. This method applies JPEG compression to input images before classification, which helps to remove subtle adversarial perturbations by introducing quantisation noise. It is especially effective against black-box attacks that rely on precise pixel-level modifications. The main configuration option is the `quality` parameter, ranging from 1 to 100. Lower values introduce more distortion and potentially higher resilience, while higher values preserve more input fidelity.

All defences save their outputs in `module4_defense_application/results/evasion/<defense_name>/`, including a `<defense_name>_results.json` file that reports metrics such as `accuracy_clean` and `accuracy_adversarial`. Additional details include per-class performance (`per_class_accuracy_clean`, `per_class_accuracy_adversarial`) and, when applicable, visual artefacts or transformed inputs. A Markdown file named `report.md` is also included to provide a human-readable summary of the defence configuration and effectiveness.

By systematically applying these tailored mitigation strategies, Module 4 transforms the theoretical recommendations from the risk analysis phase into concrete defensive actions. Each defence is executed with careful logging and structured output, ensuring that mitigation efforts are reproducible and measurable. However, the application of a defence alone does not guarantee its effectiveness. To fully understand the impact of these interventions, regarding security gains and potential side effects, `Safe-DL` proceeds to Module 3.6, where each defence is rigorously evaluated under controlled conditions.

3.6 Module 5 — Defence Evaluation

After applying the defensive strategies in Module 4, the `Safe-DL` pipeline evaluates their effectiveness through Module 5. This module depends directly on the clean baseline metrics, attack outputs from Module 2, and the post-defence results from Module 4. Its goal is to assess how much each defence mitigates its corresponding threat's effects while measuring the impact on clean data and the computational cost involved. The outcomes of this analysis are later used in Module 6 to generate the final security report, making this step a key bridge between implementation and summary.

To perform this evaluation, the framework loads three core sources: the clean baseline accuracy stored in `module2_attack_simulation/results/baseline_accuracy.json`;

the original attack metrics found under `module2_attack_simulation/results/<category>/<attack>/<attack>_metrics.json`; and the defence outputs produced in `module4_defense_application/results/<category>/<attack>/<defense>_results.json`. These inputs allow `Safe-DL`to compute standardised evaluation metrics across all threat categories.

Each defence is assessed using three scoring components: the *Mitigation Score* (MS), the *Clean Accuracy Drop* (CAD), and the *Defence Cost Score* (DCS). These are designed to jointly reflect effectiveness, generalisation, and practical feasibility.

The MS quantifies how effectively the defence reverses the impact of the attack. Its formulation depends on the attack category. For data poisoning threats, it is computed using Equation 3.2:

$$\text{MS} = \frac{\text{Accuracy}_{\text{defence}} - \text{Accuracy}_{\text{attack}}}{\text{Accuracy}_{\text{baseline}} - \text{Accuracy}_{\text{attack}}} \tag{3.2}$$

Here, $\texttt{Accuracy}_{\texttt{defence}}$ refers to the clean accuracy of the model after defence (from the `accuracy_clean` field in the defence output). $\texttt{Accuracy}_{\texttt{attack}}$ is the model's accuracy after being trained on poisoned data (from `accuracy_after_attack`), and $\texttt{Accuracy}_{\texttt{baseline}}$ corresponds to the model's original clean accuracy before any attacks (from `baseline_accuracy.json`).

For backdoor threats, the MS is calculated using Equation 3.3 and reflects the reduction in ASR:

$$\text{MS} = \frac{\text{ASR}_{\text{before}} - \text{ASR}_{\text{after}}}{\text{ASR}_{\text{before}}} \tag{3.3}$$

Here, $\texttt{ASR}_{\texttt{before}}$ denotes the ASR before any defence is applied (from `attack_success_rate`), and $\texttt{ASR}_{\texttt{after}}$ is the ASR after defence (from `asr_after_defense`).

For evasion attacks, the score reflects the improvement in adversarial accuracy relative to the theoretical maximum, as shown in Equation 3.4:

$$\text{MS} = \frac{\text{Accuracy}_{\text{defence, adv}} - \text{Accuracy}_{\text{attack, adv}}}{1 - \text{Accuracy}_{\text{attack, adv}}} \tag{3.4}$$

In this case, $\texttt{Accuracy}_{\texttt{defence, adv}}$ is the model's accuracy on adversarial examples after defence (from `accuracy_adversarial`), and $\texttt{Accuracy}_{\texttt{attack, adv}}$ is the corresponding value before defence (from `accuracy_adversarial_testset`).

The second component, the CAD, captures how much the defence affects performance on clean inputs. It is calculated using Equation 3.5:

$$\text{CAD} = 1 - \frac{\text{Accuracy}_{\text{clean, attack}} - \text{Accuracy}_{\text{clean, defence}}}{\text{Accuracy}_{\text{clean, attack}}} \tag{3.5}$$

Here, $\texttt{Accuracy}_{\texttt{clean, attack}}$ represents the clean accuracy after the attack but before defence (e.g., after poisoning or backdoor injection), and $\texttt{Accuracy}_{\texttt{clean, defence}}$ is the clean accuracy after applying the defence.

The third and final metric, the DCS, is a manually assigned value between 0.0 and 1.0 that reflects the computational or architectural overhead introduced by the defence. It is based on internal heuristics and domain knowledge.

These three metrics are integrated into a single final score using the formula in Equation 3.6:

$$\text{Final Score} = \frac{0.8 \cdot \text{MS} + 0.2 \cdot \text{CAD}}{1 + 0.1 \cdot \text{DCS}} \tag{3.6}$$

This formulation prioritises attack mitigation, moderately values clean accuracy preservation, and lightly penalises defences that impose high cost. As such, it supports robust and fair comparisons across heterogeneous defence mechanisms and attack scenarios.

All computed metrics are recorded in a JSON file at `results/defense_evaluation.json`. This file includes per-defence mitigation scores, clean accuracy drop, defence cost, and the final composite score. A human-readable Markdown file, `results/defense_evaluation_report.md`, is generated to summarise the outcomes in ranked tables and contextual explanations suitable for reporting and documentation.

This scoring framework enables a systematic and comparable evaluation of all defences across different threat categories. Balancing attack mitigation, generalisation to clean data, and implementation cost provides a defensible basis for ranking candidate strategies. The resulting scores are stored in machine-readable and human-readable formats, ensuring traceability for audit and reporting purposes. These artefacts also serve as a direct input for the next stage in the pipeline, detailed in Section 3.7, which consolidates all security assessments into a final comprehensive report.

3.7 Module 6 — Final Reporting

The final stage of the `Safe-DL`pipeline consolidates all information gathered throughout the evaluation into a single, structured Markdown report. This output is a comprehensive and traceable summary of the system's adversarial robustness, covering all stages from threat modelling to defence evaluation. It is designed to support auditing, documentation, and long-term reproducibility.

The report is generated automatically by the script `generate_final_report.py`, which traverses the enriched profile file and aggregates additional data from earlier stages of the pipeline. This file is progressively populated across Modules 1 to 6, capturing system-level metadata, executed attacks, applied defences, and corresponding evaluation scores.

In addition to the profile, the report generation script reads from several module-specific result files to ensure completeness and accuracy. These include:

- `module2_attack_simulation/results/baseline_accuracy.json`
- `module2_attack_simulation/results/<category>/<attack>/<attack>_metrics.json`
- `module3_risk_analysis/results/risk_analysis.json`

- `module4_defense_application/results/<category>/<attack>/<defense>_results.json`
- `module5_defense_evaluation/results/defense_evaluation.json`

These inputs populate key sections of the report, including system context, attack execution logs, risk assessments, defence mappings, and final evaluation metrics. The report is built using a deterministic, template-based approach, ensuring consistency across runs and full traceability of technical decisions. If a defence was skipped or failed to run, this is explicitly documented alongside explanatory notes.

The final output of this module is the file `reports/final_report.md`, which is structured into well-defined sections aligned with the stages of the pipeline. The report begins with a **System Overview**, which describes the dataset used, the model's architecture, the nature of the task, and the assumptions made about the attacker. This is followed by the **Attack Summary**, which enumerates all attacks executed in each threat category, providing brief descriptions and, where applicable, links to manipulated examples for better interpretability.

The following section, **Threat Mapping**, presents the vulnerabilities identified by the risk analysis engine in Module 3. It includes quantitative risk scores and per-class coverage statistics, clearly showing the system's susceptibility. In the **Defence Actions and Results** section, the report details the defences applied to each specific sub-attack, including their configuration, numerical evaluation metrics as derived in Module 5, and any relevant qualitative assessments such as limitations or trade-offs.

The **Global Ranking** section aggregates all evaluated defences across threat types, ordering them according to their final score to facilitate comparison and support informed decision-making. Finally, the report concludes with a section on **Post-Deployment Recommendations**, outlining general best practices for continuous monitoring, periodic re-evaluation, and maintaining adversarial robustness in real-world deployments.

Thanks to its Markdown format, the final report is human-readable and readily convertible to formats such as PDF or HTML. It also embeds or links to auxiliary artefacts produced throughout the pipeline, including visual aids such as filtered samples, confusion matrices, or anomaly scores. This report closes the `Safe-DL` pipeline with a transparent and reproducible artefact by aggregating all results and decisions into a cohesive document. It supports technical insight and organisational accountability, offering a solid foundation for future reassessment, certification efforts, or targeted system hardening.

3.8 Chapter Summary

This chapter detailed the internal architecture and operational flow of the `Safe-DL` pipeline for adversarial robustness evaluation in DL. The system is organised into six sequential modules, each responsible for a distinct phase of the workflow: threat modelling, attack simulation, risk analysis, defence application, defence evaluation, and final reporting.

Module 1 introduces the system context by gathering metadata and defining the relevant threat assumptions. Building on that foundation, Module 2 simulates attacks tailored to the selected threat model, providing concrete evidence of potential vulnerabilities. Module 3 then analyses these results to identify which system components are most at risk, quantifying exposure at the class level and informing mitigation priorities. Module 4 responds to these findings by applying appropriate defensive strategies selected and configured automatically according to the threat profile. Module 5 evaluates how effective each defence is, using a scoring scheme that balances attack mitigation, clean accuracy preservation, and implementation cost. Finally, Module 6 brings everything together in a structured Markdown report summarising the system's security posture and offering a complete audit trail of all decisions and results.

These components enable a reproducible, modular, and security-conscious evaluation process. The framework supports rigorous testing under diverse adversarial conditions and provides stakeholders with transparent evidence of both model vulnerabilities and the effectiveness of applied defences. Chapter 4 builds upon this foundation by presenting a set of real-world case studies, demonstrating how `Safe-DL` performs across different datasets, architectures, and threat scenarios.

Chapter 4

Experimental Validation and Framework Discussion

Having detailed the internal design, capabilities, and outputs of each pipeline stage in Chapter 3, this chapter now puts the `Safe-DL` framework into practice through a complete use case scenario. The goal is to validate the framework holistically, ensuring that each module operates correctly in isolation and functions coherently as an integrated workflow.

The validation covers all phases of the pipeline, from the initial threat modelling to the generation of the final security report, illustrating how the system behaves in a real-world context. This process serves as a technical verification and a practical demonstration of the results' usability, automation robustness, and interpretability.

The chapter also showcases how `Safe-DL` can support secure machine learning development, especially for users lacking deep expertise in adversarial techniques, by grounding the evaluation in a realistic task and dataset. The discussion combines quantitative results from the framework's output artefacts with qualitative observations about its modularity, user experience, and diagnostic value.

4.1 Experimental Setup

This section outlines the experimental setup used to validate the `Safe-DL` framework in a complete end-to-end execution. The objective is to evaluate whether the framework can effectively identify relevant threats, simulate impactful attacks, deploy meaningful defences, and produce interpretable reports under conditions that reflect practical deployment scenarios.

Instead of relying on abstract parameter choices or synthetic tasks, the validation is anchored in a realistic use case that motivates all downstream decisions. This includes the selection of dataset, model architecture, and threat configuration, all aligned with the simulated deployment's operational and security context.

What follows is a description of this scenario, the motivations behind the experimental choices, and the evaluation methodology used to assess the technical and practical performance of `Safe-DL`.

4.1.1 Simulated Scenario and Use Case

To motivate the application of the `Safe-DL` framework, we simulate a realistic use case involving a mid-sized manufacturing company named **VisiTech Circuits**, which integrates AI into its internal quality control pipeline. The company produces consumer electronics and relies on image-based inspection to detect visual defects such as cracks, misalignments, or soldering faults on assembled printed circuit boards (PCBs).

The engineering team has developed a computer vision model that classifies PCB images as "defective" or "non-defective." This model is embedded into a desktop application operated locally by technicians on the factory floor. The system runs entirely offline without exposure to public APIs or external network interfaces.

Although the team is experienced in developing and deploying machine learning models, they lack expertise in adversarial machine learning and AI security. Internal concerns over robustness and auditability, especially after including third-party datasets and upcoming compliance reviews, lead them to proactively assess the model's security posture before deployment.

To that end, they adopt the `Safe-DL` framework with the expectation of receiving a high-level overview of threats relevant to their context, simulation of plausible adversarial attacks, automatic selection and evaluation of suitable defences, and a final report that consolidates all findings in a format accessible to both technical and non-technical stakeholders.

This scenario underpins the experimental setup presented in this chapter, shaping the choice of dataset, model architecture, and evaluation strategy to mirror real-world deployment conditions and security motivations.

4.1.2 Dataset, Model Architecture, and Execution Environment

To simulate the previous deployment scenario, the dataset and model architecture were selected from the built-in options natively supported by the `Safe-DL` framework. These choices were guided by the need to realistically capture relevant security threats while maintaining tractability, reproducibility, and ease of interpretation.

The dataset used in this evaluation is CIFAR-10, which contains 60,000 colour images of size 32×32 pixels, evenly distributed across 10 distinct classes such as airplanes, birds, and cars. It is a well-established benchmark for image classification tasks and offers multiple advantages for this study. First, its visual nature facilitates qualitative inspection of adversarial artefacts, including perturbations and poisoned inputs. Second, its widespread adoption enables the results to be contextualised within the broader literature on adversarial robustness. Third, it is fully compatible with the attack and defence modules implemented in `Safe-DL`, allowing seamless integration and minimising engineering effort. Finally, the classification task it represents closely mirrors the type of visual inspection performed in the simulated use case, making it a natural choice for the scenario previously outlined.

Although `Safe-DL` supports other datasets like MNIST, SVHN, or custom user-defined inputs, CIFAR-10 was chosen for its balance between complexity, relevance, and interpretability.

The classification model adopted for this evaluation is ResNet18, a CNN composed of 18 layers with residual skip connections. Introduced by He et al. (2015), this architecture is known for addressing the degradation problem in deep networks by facilitating gradient propagation during training. ResNet18 is widely used as a baseline in robustness literature, providing a reliable reference point for comparative evaluation. Its moderate depth and low computational overhead make it well-suited for real-time applications and embedded systems, reflecting the simulated deployment's resource constraints and reliability requirements. In addition, ResNet18 is fully supported within `Safe-DL`, requiring no additional wrappers or configuration steps.

The model was trained from scratch on CIFAR-10 using standard data augmentation techniques, including random horizontal flipping and per-channel normalisation. A fixed random seed (`42`) was used consistently across all modules to guarantee deterministic behaviour and facilitate reproducibility.

All experiments were conducted on a workstation equipped with two Intel Xeon 6258R CPUs @ 2.7 GHz, 128 GB of RAM, and an NVIDIA Quadro RTX 8000 GPU with 48 GB of VRAM. This environment allowed smooth execution of all stages in the `Safe-DL` pipeline, including compute-intensive tasks such as adversarial training and feature-based risk analysis, without compromising runtime or consistency. While the framework is compatible with lighter hardware configurations, this setup ensured that no constraints were imposed on the evaluation by infrastructure limitations.

4.1.3 Evaluation Criteria

A multi-dimensional evaluation strategy was adopted to assess the effectiveness and usability of the `Safe-DL` framework. The goal is to measure technical robustness against adversarial threats and determine whether the system offers meaningful guidance, runs reliably across all stages, and generates insights that support informed decision-making.

A successful run is one where the complete pipeline executes, from threat modelling to final report generation, without critical errors or manual intervention. Each module must function coherently. The initial YAML-based profile (`profile.yaml`) must be correctly parsed and translated into meaningful attack and defence configurations. Simulated attacks must result in observable degradations of model performance, and the applied defences should effectively mitigate adversarial effects while preserving clean accuracy. The framework is expected to generate structured outputs in both `.json` and `.md` formats. These outputs serve as the basis for traceability, reproducibility, and final decision reporting, ultimately culminating in the `final_report.md` file aggregating all results, metrics, and justifications.

From a quantitative perspective, the evaluation is grounded in automatic metrics computed across different pipeline stages. In Module 2, the impact of each adversarial threat is captured using the ASR, which measures the proportion of adversarial inputs that successfully cause misclassification and the adversarial accuracy, which reflects the model's performance on perturbed inputs. For poisoning-based attacks, an additional metric, the drop in clean validation accuracy,

is used to quantify how much the poisoned training data affects generalisation. Module 3 computes a risk score for each threat derived from its likelihood (based on attack success) and severity (based on its deviation from baseline performance). This enables the prioritisation of vulnerabilities that pose the greatest practical danger. In Module 4, defences are applied, and their impact is re-evaluated using the same metrics from Module 2, allowing direct measurement of threat mitigation. Finally, Module 5 aggregates these results into a Final Score, which combines mitigation effectiveness, clean accuracy preservation, and computational or architectural cost into a single normalised value. This composite score facilitates direct comparison across defence strategies, enabling users to weigh trade-offs between robustness, utility, and cost in a principled and interpretable way.

Beyond raw performance, the framework is also evaluated regarding usability and transparency. The use of YAML profiles promotes reproducibility and precise specification of threat contexts. The CLI provides intelligent defaults and interactive guidance, helping non-expert users navigate each stage confidently. All reports include both raw metrics and human-readable summaries, making results accessible to a broad range of stakeholders. This blend of automation, explainability, and modularity positions `Safe-DL` as a practical and trustworthy tool for deploying secure machine learning systems in real-world environments.

With the scenario, dataset, model, and evaluation strategy firmly defined, the experimental setup outlined in this section establishes a concrete and realistic foundation for validating the `Safe-DL` framework. These choices ensure the evaluation reflects practical deployment conditions and adversarial concerns relevant to modern machine learning pipelines. In the following Section 4.2, we execute the full `Safe-DL` pipeline and examine its behaviour across all stages, highlighting the quantitative outcomes and qualitative observations arising from this end-to-end validation.

4.2 Framework Execution

This section illustrates the practical execution of the `Safe-DL` framework in a simulated deployment setting. The goal is to demonstrate how each module operates within the pipeline to detect, assess, and mitigate adversarial threats in a fully automated and auditable manner.

The framework is executed sequentially, with each module receiving structured input, applying predefined logic, and generating reproducible outputs. From threat modelling to final reporting, the system transitions through all stages without manual intervention, enabling a complete evaluation of its technical robustness and operational usability.

Each subsection presents one module in detail, describing its objectives, configuration, and results. Emphasis is placed on the internal mechanics of each step and the quality of the generated artifacts, including attack simulations, risk assessments, defence strategies, and final reports. This step-by-step walk-through validates the system's functional integrity while showcasing its practical utility in real-world security workflows.

4.2.1 Module 1 — Threat Modelling

The first module in the `Safe-DL` pipeline initiates the adversarial risk assessment process by constructing a formal threat model through an interactive questionnaire. This stage defines the system's security assumptions, operational context, and potential attack surface. In our evaluation, the responses were tailored to the VisiTech scenario.

The threat modelling was conducted via the CLI-driven script `run_module1.py`, which prompted the user to answer a series of structured questions covering critical dimensions: adversary capabilities, model architecture, data provenance, and deployment environment. The resulting profile characterised the attacker as operating under a *gray-box* access model, lacking direct access to weights but capable of reverse engineering or leveraging side-channel information. The attack goal was set to *targeted*, assuming the adversary aims to misclassify specific defective parts as functional.

The deployment scenario was defined as *on-device*, with the model integrated into desktop-based inspection software and accessed through a local graphical interface, denoted in the profile as *local_app*. The training data was considered to have *medium* sensitivity and originated from a combination of proprietary defect imagery and third-party image repositories, classified as a *mixed* source. The model architecture specified was a CNN, aligning with the actual use of ResNet18 in this use case.

The system automatically selected `data_poisoning` and `backdoor_attacks` as relevant threats based on these inputs. The user additionally enabled `evasion_attacks` to account for test-time robustness concerns. Once finalised, the configuration was serialised into the profile file `visitech.yaml`, which acts as the canonical threat profile for all downstream modules.

Figure 4.1: Generated threat profile `visitech.yaml`.

```yaml
name: visitech
threat_model:
  attack_goal: targeted
  data_sensitivity: medium
  deployment_scenario: on_device
  interface_exposed: local_app
  model_access: gray-box
  model_type: cnn
  training_data_source: mixed
  threat_categories:
    - data_poisoning
    - backdoor_attacks
    - evasion_attacks
```

This file, stored in the `profiles/` directory, is automatically parsed by subsequent modules

to ensure consistency across attack simulation, defence selection, and risk evaluation. Since all answers were provided via CLI, the process remains reproducible, auditable, and adaptable, allowing automation and expert refinement.

Screenshots of the CLI interaction that produced this file are available in Appendix B, Section B.1. Figures B.1 and B.2 illustrate the complete execution trace.

This module translates informal threat assumptions into a structured, machine-readable format, enabling the `Safe-DL` pipeline to perform context-aware security evaluations in all subsequent stages.

4.2.2 Module 2 — Attack Simulation

The second module of the `Safe-DL` framework simulates the adversarial threats identified during the threat modelling process. Using the YAML profile defined in the previous step, this module selects appropriate attack strategies, configures the experimental environment, and executes various attack techniques against the model under evaluation. The objective is to evaluate the model's robustness by quantifying performance degradation under realistic threat vectors.

The simulation process begins with an automated configuration phase, initiated by running the script `setup_module2.py`. This script parses the `visitech.yaml` profile, prompts the user to select the dataset, model architecture, and attack configurations, and stores the resulting setup in a structured format that ensures reproducibility and traceability.

In the context of our evaluation, the dataset selected was CIFAR-10, and the model architecture was ResNet18. These components are available as native options in `Safe-DL` and were consistent with the threat modelling assumptions provided earlier.

Upon loading the threat profile, the framework identified three relevant attack categories: data poisoning, backdoor injection, and evasion attacks. For each of these, `Safe-DL` proposed default configurations based on the threat context, which were accepted without modification for this evaluation. These configurations were then automatically stored in the updated YAML profile for reproducibility and later use.

For the *data poisoning* scenario, a Label Flipping strategy was applied using a many-to-one mapping. Specifically, 8% of the training samples had their labels flipped to class 9 (`truck`), with the source classes selected randomly rather than fixed.

The framework configured a Static Patch attack in the case of *backdoor injection*. A white square trigger was embedded in the bottom-right corner of 15% of training images, and after it, all of them were relabelled to class 7 (`horse`). The patch had a relative size of 0.15 and was fully visible (blend alpha set to 1.0). The label mode was set to corrupted, ensuring the trigger-target association during training.

For *evasion attacks*, two methods were selected. The first was the PGD attack, configured with $\varepsilon = 0.03$, a step size $\alpha = 0.01$, and 50 iterations with random initialisation enabled. The second was the SPSA attack, using an ε of 0.03, a delta of 0.01, a learning rate of 0.01, and 150 optimisation steps. The attack was run on a batch size of 32 and limited to 500 samples to ensure tractability.

Figure 4.2: Excerpt of updated `visitech.yaml` with attack override configurations.

```yaml
attack_overrides:
  backdoor:
    static_patch:
      blend_alpha: 1.0
      label_mode: corrupted
      patch_position: bottom_right
      patch_size_ratio: 0.15
      patch_type: white_square
      poison_fraction: 0.15
      target_class: 7
  data_poisoning:
    label_flipping:
      flip_rate: 0.08
      source_class: null
      strategy: many_to_one
      target_class: 9
```

Once all parameters were confirmed, they were automatically recorded in the updated YAML configuration file. This updated profile captures all attack-specific overrides and ensures that future executions remain consistent and auditable. Figure 4.2 shows an excerpt of this file, highlighting how each attack category was configured in a structured, machine-readable format.

This configuration process ensures alignment between the attack parameters and the previously defined threat model. It also enforces consistency, reproducibility, and traceability for the following attack simulation phase.

After finalising the configuration, the adversarial simulation was executed by running the script `run_module2.py`. This phase began with training a clean ResNet18 model on CIFAR-10, which achieved a baseline test accuracy of **82.93%**. This serves as a reference point to assess the impact of each configured attack.

The Label Flipping attack introduced label noise into 8% of the training set, resulting in a significant performance drop, reducing clean accuracy to **65.78%**. This illustrates how relatively small perturbations in the training labels can compromise generalisation.

In the case of the Static Backdoor injection, the clean accuracy decreased to **69.40%**, while the model exhibited an ASR of **3.00%**. This indicates the presence of a hidden vulnerability, where inputs containing the trigger pattern are misclassified into the attacker-chosen target class.

For white-box evasion, the PGD attack proved especially effective. Under adversarial conditions, the model's accuracy dropped was **19.10%**, demonstrating the high susceptibility of the model to imperceptible input perturbations.

Finally, the black-box SPSA attack also produced a notable degradation. Despite lacking gradient access, it reduced adversarial accuracy to **29.20%**, reinforcing the feasibility of real-world evasion attacks under limited attacker knowledge.

A consolidated summary of these results is provided in Table 4.1. These metrics serve as the quantitative foundation for the risk prioritisation and defence selection in the next module.

Table 4.1: Impact of adversarial attacks. Clean accuracy is shown before and after attack; the third metric reflects the attack's effectiveness.

Scenario	Clean Acc. (%) Before	Clean Acc. (%) After	Attack Metric (%)
Label Flipping	82.93	65.78	65.78 (Degraded Acc.)
Static Patch	82.93	69.40	3.00 (ASR)
PGD	82.93	82.93	19.10 (Adv. Acc.)
SPSA	82.93	82.93	29.20 (Adv. Acc.)

Each attack also produced a set of result artifacts that enhance traceability and analysis. These include structured reports in Markdown format (`*.md`) and image-based visualisations of adversarial examples. Screenshots of the whole execution trace from `run_module2.py` are available in Figures B.7 to B.12, providing a step-by-step view of the pipeline's automated operations.

A full Markdown report for the label flipping attack is included as Listing B.1. Additionally, visual samples of the generated attacks, including poisoned images, backdoor triggers, and adversarial perturbations, are shown in Figures B.13 to B.15. These resources improve interpretability and reproducibility, allowing future users or auditors to replicate the setup and verify outcomes.

With this simulation phase completed, the framework has successfully exposed the vulnerabilities defined by the threat model and recorded the results in a structured and interpretable manner. The next step focuses on analysing these risks and guiding defence selection, as discussed in Module 3.

4.2.3 Module 3 — Risk Analysis

The third module of the `Safe-DL` framework transforms the raw results obtained from the simulated attacks into structured risk assessments. This step is essential for prioritising threats not only by their technical impact but also by their practical feasibility and detectability within the deployment scenario.

Each attack is evaluated along three independent dimensions: *severity*, *probability*, and *visibility*. Severity captures the extent to which the model's performance is degraded, probability reflects the likelihood of the attack succeeding in the specified context, and visibility assesses how easily the attack could be detected through human inspection or lightweight anomaly detection. These components are combined using a risk scoring heuristic, shown in Equation 4.1, which gives higher weight to effective and stealthy attacks.

$$\text{Risk Score} = \text{Severity} \times \text{Probability} \times (1 + (1 - \text{Visibility})) \tag{4.1}$$

Table 4.2 presents the numerical values computed for each attack simulated in Module 2. These scores form the basis for prioritisation and defence selection in the following stages.

In addition to the numerical metrics, a qualitative risk matrix helps visualise each threat's severity and probability relationship. Table 4.3 provides this mapping, where attacks with high severity and high probability, such as PGD and SPSA, appear in the most critical quadrant.

Table 4.2: Computed risk scores and intermediate metrics per attack.

Attack	Type	Severity	Probability	Visibility	Risk Score
Label Flipping	Data Poisoning	0.572	1.00	0.624	0.787
Static Patch	Backdoor	0.336	1.00	0.600	0.470
PGD	Evasion	1.000	1.00	0.300	1.700
SPSA	Evasion	1.000	0.80	0.200	1.440

Table 4.3: Qualitative risk matrix showing severity versus probability.

Severity \ Probability	Low	Medium	High
Low	–	–	–
Medium	–	–	Label Flipping, Static Patch
High	–	–	PGD, SPSA

Based on the computed risk scores, the attacks were ranked in descending order of criticality: PGD (1.70), SPSA (1.44), Label Flipping (0.79), and Static Patch (0.47). For each threat, `Safe-DL` automatically suggested suitable mitigation strategies based on their characteristics and practical feasibility. These recommendations included Data Cleaning and Per-class Accuracy Monitoring for Label Flipping, Activation Clustering and Spectral Signature analysis for Static Patch attacks, Adversarial Training and Randomized Smoothing for PGD, and Gradient Masking and JPEG-based Preprocessing for SPSA. While these suggestions were recorded in the YAML profile to support the next module, their application remained subject to user confirmation and configuration during the defence setup phase.

To support transparency and auditability, all outputs from Module 3 were saved in structured formats. The files `risk_analysis.json` and `risk_report.md`, located in the `results/` directory, contain the complete assessment. Additionally, Figure B.16 in Appendix B, Section B.3, illustrates the CLI trace of this process.

This risk analysis phase is crucial in transforming raw attack outcomes into structured and prioritised insights. It closes the diagnostic portion of the pipeline while setting the stage for the selection and application of targeted defences in the next module.

4.2.4 Module 4 — Defence Application

Building upon the risk prioritisation performed in Module 3, Module 4 of the `Safe-DL` framework is responsible for applying mitigation strategies tailored to each identified threat. The framework automatically suggests appropriate defences by mapping attack types and risk scores to a curated library of defence techniques grounded in academic best practices. Users are presented with these suggestions through a CLI and can accept the defaults or manually override any configuration to reflect domain-specific constraints or preferences.

The defence setup process begins with the execution of `setup_module4.py`, which parses the threat profile file (`visitech.yaml`) and loads the previously simulated attack scenarios. Based on these, the system recommends one or more defences per attack. These defences span multiple families, including input preprocessing such as JPEG compression, data sanitisation methods like loss-based filtering, robust training techniques including adversarial training, and model-level anomaly detection strategies such as activation clustering and spectral signature analysis.

In our evaluation, the default suggestions were accepted in most cases. However, for the label flipping scenario, a manual override was performed to restrict mitigation to a single technique (`data_cleaning`). The whole configuration process was executed through the CLI, and a complete trace is shown in Appendix B, Section B.4, Figures B.17 to B.19.

Each defence and its corresponding parameters were saved in the updated profile file for traceability and reproducibility. For instance, in the case of Label Flipping, the selected method was Data Cleaning with a confidence threshold of 0.9. For Static Patch backdoors, two defences were selected: Activation Clustering, configured with 2 clusters, and Spectral Signatures, using a filtering threshold of 0.9. Against the PGD evasion attack, the framework applied Adversarial Training, which used FGSM examples with $\varepsilon = 0.03$, and Randomised Smoothing with a noise level of $\sigma = 0.25$. For the SPSA attack, both Gradient Masking, with a masking strength of 0.5, and JPEG preprocessing, set to a quality level of 75, were enabled.

All selected configurations were recorded in the YAML profile, as shown in the excerpt in Figure 4.3. This centralised specification ensures that defence strategies remain consistent across executions and are fully auditable.

Figure 4.3: Excerpt of updated `visitech.yaml` containing defence configurations.

```yaml
defense_config:
  backdoor:
    static_patch:
      activation_clustering:
        num_clusters: 2
      defenses:
      - activation_clustering
      - spectral_signatures
      spectral_signatures:
        threshold: 0.9
  data_poisoning:
    label_flipping:
      data_cleaning:
        method: loss_filtering
        threshold: 0.9
      defenses:
      - data_cleaning
```

This configuration phase guarantees that each selected defence is aligned with the specific threats and associated risk scores. The setup process ensures both contextual relevance and full

reproducibility by supporting interactive customisation and automatically updating the profile file.

Following this setup, the framework advances to the mitigation stage, where each configured defence is applied and its effectiveness rigorously evaluated under adversarial conditions. This is orchestrated by executing the `run_module4.py` script, which retrains the model accordingly and records all outputs, including models, metrics, logs and visual artifacts, in structured `.json` and `.md` formats to support downstream analysis and auditability.

The results of the defence evaluation reveal substantial variation in effectiveness depending on the strategy applied and the nature of the underlying threat.

In the case of Label Flipping, the Data Cleaning method proved to be a lightweight yet effective defence. By removing **6 967** suspicious training samples, it improved the clean accuracy from **65.78%** (under attack) to **68.37%**. While the gain is modest, it demonstrates that even simple data sanitisation heuristics can partially recover model performance with minimal overhead or complexity.

For the Static Patch backdoor, two anomaly detection defences were tested. The first, Activation Clustering, achieved excellent results. It isolated **10 562** likely poisoned examples and brought the ASR down to **0.088%** while maintaining a clean accuracy of **61.54%**. This indicates that clustering-based approaches can effectively identify and mitigate backdoor triggers with limited performance trade-offs.

The second method, Spectral Signatures, was less successful. Despite removing an even larger number of samples (**19 529**), it degraded clean performance to **51.58%** and, unexpectedly, increased the ASR to **3.76%**, above the baseline of **3%** observed without any defence. Results suggest that the technique failed to isolate the backdoor features, resulting in collateral damage to benign samples and ineffective mitigation.

For the PGD evasion attack, Adversarial Training proved to be the most effective defence overall. It raised adversarial accuracy from **19.10%** to **41.88%** while only slightly reducing clean accuracy to **76.55%**. This reflects the known trade-off between robustness and accuracy and validates adversarial training as a strong baseline defence.

Conversely, Randomised Smoothing, another robustness-oriented technique, underperformed in this setup. Although it added stochastic noise during inference, it failed to meaningfully improve adversarial accuracy (only **19.88%**) and severely harmed clean accuracy, which dropped to **37.20%**. This suggests that, without careful tuning, smoothing-based methods may do more harm than good.

Against the SPSA attack, gradient masking had no measurable impact on adversarial robustness. The adversarial accuracy remained at **29.60%**, nearly identical to the value observed without any defence. Clean accuracy also stayed unchanged at **82.80%**. These results indicate that this defence neither improved robustness nor introduced any degradation, ultimately providing no meaningful protection in this setting.

In contrast, JPEG preprocessing significantly degraded image quality. Clean accuracy dropped to just **11.94%**, and no improvement in adversarial accuracy was observed. This demonstrates that

aggressive input transformation techniques, when not properly tuned, can severely compromise model performance without yielding security benefits.

A consolidated summary of outcomes is presented in Table 4.4, enabling direct comparison of defence strategies. The clean accuracy is reported for all scenarios, while adversarial accuracy is shown only for evasion-based threats, and the ASR is reported exclusively for backdoor attacks.

Table 4.4: Effectiveness of defence strategies. Clean accuracy is reported post-defence; adversarial accuracy and ASR are shown where applicable.

Defence Strategy	**Clean Acc. (%)**	**Adv. Acc. (%)**	**ASR (%)**
Data Cleaning (Label Flipping)	68.37	–	–
Activation Clustering (Static Patch)	61.54	–	0.088
Spectral Signatures (Static Patch)	51.58	–	3.76
Adversarial Training (PGD)	76.55	41.88	–
Rand. Smoothing (PGD)	37.20	19.88	–
Gradient Masking (SPSA)	82.80	29.60	–
JPEG Preprocessing (SPSA)	11.94	29.40	–

Each defence also produced structured artifacts, including configuration metadata, performance summaries, and visual logs. These were stored in both machine-readable (`.json`) and human-readable (`.md`) formats. Appendix B, Section B.4 includes the full CLI trace for `run_module4.py` in Figures B.20 to B.26, as well as sample outputs for image filtering in Figures B.28 and B.27.

To illustrate the reporting format, the complete Markdown summary for the Data Cleaning defence is provided in Listing B.2, showing the removed samples and their attributes. These artifacts reinforce the transparency and reproducibility of `Safe-DL` by documenting outcomes and the rationale behind each mitigation decision.

These results underscore the trade-offs between robustness and model utility, emphasising the importance of context-aware defences. The following module builds upon these outcomes by assigning structured scores to each mitigation strategy, enabling a comparative evaluation of their effectiveness.

4.2.5 Module 5 — Defence Evaluation

Module 5 of the `Safe-DL` pipeline focuses on evaluating and ranking the defence strategies applied in Module 4. This evaluation uses a unified scoring procedure that balances robustness gains against clean accuracy degradation and computational overhead. For each active defence, four key metrics are computed: the **Mitigation Score**, the **Clean Accuracy Drop**, the **Defence Cost Score**, and the resulting **Final Score**, which summarises overall effectiveness.

To execute this stage, we ran the script `run_module5.py`, which parsed the `visitech.yaml` profile and automatically evaluated all configured defences. The results were saved in both machine-readable (`.json`) and human-readable (`.md`) formats, enabling downstream reporting and traceability.

Table 4.5 presents a quantitative comparison of all applied defence strategies:

Table 4.5: Evaluation metrics for all defence strategies.

Attack	**Defence**	**Mitigation**	**CAD**	**Cost**	**Final Score**
Static Patch	Activation Clustering	0.970	0.887	0.300	0.926
	Spectral Signatures	0.000	0.743	0.500	0.142
Label Flipping	Data Cleaning	0.151	1.039	0.200	0.322
PGD	Adversarial Training	0.282	0.923	0.800	0.380
	Randomised Smoothing	0.010	0.449	0.500	0.093
SPSA	Gradient Masking	0.006	0.998	0.400	0.196
	JPEG Preprocessing	0.003	0.144	0.100	0.031

The evaluation results reveal a diverse spectrum of effectiveness among the tested defences in terms of robustness gains and utility preservation. The most successful strategy was Activation Clustering, which achieved a mitigation score of **0.970** by effectively reducing the ASR in the static patch backdoor scenario. It maintained a relatively high clean accuracy (CAD of **0.887**) while incurring only moderate computational cost, leading to the best overall score of **0.926**.

In contrast, Spectral Signatures, also applied to the same attack, failed to isolate the poisoned samples. It achieved a mitigation score of **0.000**, degraded clean performance (CAD **0.743**), and resulted in a low final score of **0.142**. This indicates that the technique introduced noise without effective filtering.

For the label flipping attack, Data Cleaning offered a modest mitigation score of **0.151**, successfully identifying and removing noisy samples. It slightly improved the model's clean accuracy post-attack, as reflected by a CAD score of **1.039**. Combined with a low defence cost of **0.200**, this yielded a final score of **0.322**, reflecting a reasonable trade-off given its simplicity.

Regarding evasion attacks, Adversarial Training proved the most effective against PGD. It boosted adversarial robustness significantly (mitigation score **0.282**) while slightly compromising clean accuracy (CAD **0.923**). However, due to its high-cost score (**0.800**), the final score reached only **0.380**, highlighting the robustness-performance-cost trade-off inherent to retraining-based strategies.

Randomised Smoothing, also applied to PGD, produced poor outcomes across all dimensions. Its mitigation score was merely **0.010**, clean accuracy degraded severely (CAD **0.449**), and the resulting final score was only **0.093**, indicating that the defence added noise without meaningful gains.

For the SPSA attack, Gradient Masking preserved both clean and adversarial accuracy but failed to enhance robustness meaningfully, with a mitigation score of just **0.006** and final score **0.196**. Finally, JPEG Preprocessing was the weakest strategy overall: its mitigation score was a

negligible **0.003**, and the clean accuracy degradation was drastic (CAD **0.144**), yielding the lowest final score of **0.031**.

The CLI execution trace for this module is included in Appendix B, Section B.5, Figures B.29 and B.30. Full output files, such as `defense_evaluation.json` and `defense_evaluation_report.md`, are stored in the `results/` directory and provide structured, machine-readable summaries for transparent comparison and reproducibility.

These results are a foundation for the final reporting module, consolidating all evaluations into a unified and auditable documentation artifact.

4.2.6 Module 6 — Final Reporting

The final module of the `Safe-DL` pipeline consolidates the outputs of all previous stages into a comprehensive and structured report. This reporting phase serves a dual purpose: it acts as a centralised audit trail for all security-related processes. It also provides stakeholders with a clear overview of the system's vulnerabilities, applied mitigations, and recommended actions moving forward.

The generated report includes a detailed overview of the evaluated model and dataset, describing architectural characteristics and input specifications. It also incorporates the complete threat profile defined during Module 1, including adversarial assumptions, attacker goals, access levels, and the deployment scenario. From Module 2, the report summarises how each simulated attack affected system performance, referencing detailed markdown reports for in-depth analysis.

Risk assessment results from Module 3 are included next, featuring per-attack risk scores, the qualitative risk matrix, and a ranked list of threats based on their computed severity, probability, and visibility. Following this, the report details which defences were applied in Module 4 and their impact on clean and adversarial accuracy. Module 5 contributes a ranked evaluation of each defence strategy, presenting the mitigation score, clean accuracy drop, computational cost, and final effectiveness score.

The report concludes with an executive summary highlighting the most critical risks identified throughout the pipeline and the most effective defence strategies observed during experimentation. Additionally, it provides forward-looking guidance on integrating adversarial robustness into post-deployment stages, with suggestions for security checks within CI/CD pipelines, data drift monitoring, incident response protocols, and periodic auditing of deployed models.

After executing the final reporting script, the framework produces two core artifacts: a complete Markdown report stored as `src/reports/final_report.md`, suitable for technical review or integration into documentation workflows, and a command-line trace confirming the successful generation of the report. These outputs are included in Appendix B, Section B.6, with Figure B.31 showing the CLI execution and Listing B.3 presenting the full contents of the final report.

Module 6 demonstrates how `Safe-DL` delivers a fully integrated, end-to-end pipeline for adversarial risk management in DL systems, providing transparency and actionable insights for secure deployment.

The completion of the six modules presented in this section illustrates the full execution of the `Safe-DL` pipeline, from threat modelling and attack simulation to defence application and final reporting. Each step contributed to a layered understanding of the system's vulnerabilities and capacity for mitigation under diverse adversarial scenarios. By maintaining a structured YAML-based configuration and producing reproducible artifacts at every stage, the framework ensures both traceability and adaptability to new threat contexts. With the technical implementation complete, we now turn to a broader reflection on the results, limitations, and implications of the pipeline in Section 4.3.

4.3 Discussion and Critical Analysis

This section critically examines the `Safe-DL` framework in light of its design goals, practical execution, and observed performance across the full security lifecycle. Beyond presenting empirical results, the aim is to assess whether the framework delivers on its promise of making adversarial robustness more actionable, traceable, and usable for real-world deployments.

To support this analysis, we revisit the research objectives and goals stated in Section 1.4, evaluate the effectiveness of defences across different threat types, and reflect on broader usability factors such as transparency, automation, and alignment with emerging AI regulations. A comparative review with related frameworks is also presented to contextualise the contributions of `Safe-DL` within the adversarial ML landscape. Finally, we identify the main limitations of the current prototype and report the engineering challenges faced during its development.

Rather than treating modules in isolation, this discussion takes a holistic view of the pipeline, emphasising how its components interact to support principled and reproducible security assessments for DL models.

4.3.1 Critical Alignment with Stated Goals

This subsection revisits this dissertation's general and specific objectives, research questions, and hypotheses, evaluating to what extent the implementation and experimental results of `Safe-DL` fulfilled the dissertation's intended contributions. Each design goal is examined in light of the corresponding functionality within the framework and the empirical evidence gathered during the evaluation.

Starting with threat modelling and contextualisation (Objective 1), the interactive YAML-based configuration system successfully guided users through key aspects of the threat landscape, including access level, deployment scenario, and adversarial goals. This interface directly addressed Research Question 1 by enabling non-expert users to construct valid threat models without requiring deep expertise in adversarial machine learning. Moreover, Hypothesis 3, which anticipated that the configuration and reporting design would be usable and auditable, was confirmed through the reproducible YAML profiles and Markdown outputs that served as the backbone of the entire pipeline.

The simulation of adversarial attacks (Objective 2) was addressed by integrating diverse threat types, including evasion (e.g., PGD, SPSA), data poisoning, and backdoors. This diversity provided a comprehensive view of vulnerabilities, fulfilling the intent of Research Question 2. Each attack was executed in a controlled and reproducible environment, confirming Hypothesis 1, which proposed that `Safe-DL` would reliably identify adversarial vulnerabilities across multiple threat types.

Objective 3 focused on risk analysis and impact quantification. Module 3 delivered this through risk scoring algorithms that combined clean accuracy degradation, ASR, and other impact metrics into interpretable risk scores. These quantitative insights supported the prioritisation of threats and were directly used in defence selection logic, aligning well with Research Question 3. This structured analysis also contributed to the framework's explainability and transparency.

The automation of defence selection and configuration (Objective 4) addressed Research Question 4 by mapping simulated threat profiles to suitable mitigation strategies. These defences, including adversarial training, randomised smoothing, spectral signatures, and activation clustering, were automatically configured based on the risk matrix and stored in the updated profile. Although users retained override capabilities, the default recommendations were effective in most cases, supporting Hypothesis 2, which stated that automated defence selection could achieve a meaningful trade-off between robustness, usability, and cost.

Objective 5, which concerned defence evaluation, was implemented in Module 5 using composite metrics such as Mitigation Score, Clean Accuracy Drop, Defence Cost, and Final Score. These enabled consistent comparisons between strategies and directly answered Research Question 5. The scoring methodology helped expose trade-offs between defence strength and performance degradation, guiding users in making informed security decisions.

The final objectives (6 and 7) emphasised reporting quality, reproducibility, and extensibility. Throughout the pipeline, all artifacts, including evaluation metrics, configuration files, and human-readable reports, were stored in structured formats such as JSON, YAML, and Markdown. This fulfilled Research Question 6 by ensuring auditability and transparency and demonstrated compliance-readiness with AI regulations like the *AI Act*. Furthermore, the modular design of `Safe-DL` confirms the feasibility of integrating new attack categories in the future, satisfying Objective 7 regarding future extensibility.

In summary, the evaluation of `Safe-DL` confirms that the framework achieved its primary design goals and offers strong support for its guiding hypotheses. While limitations remain (see Section 4.3.5), the framework serves as a proof-of-concept for a usable, extensible, and auditable pipeline that guides security assessments across diverse DL applications.

4.3.2 Effectiveness Across Threat Categories

This subsection analyses the effectiveness of the `Safe-DL` framework in mitigating adversarial threats across three major categories: backdoors, data poisoning, and evasion attacks. The discussion is grounded in the evaluation methodology introduced in Section 4.1.3, with a focus on four quantitative metrics: MS, CAD, DCS, and the composite Final Score computed in Module 5.

Regarding backdoor threats, the framework evaluated two defences against a static patch attack: *Activation Clustering* and *Spectral Signatures*. Activation Clustering proved highly effective, achieving an MS of 0.97, a CAD of 0.887, and a low cost of 0.3 — resulting in a Final Score of 0.926, the highest among all evaluated defences. In contrast, Spectral Signatures failed to reduce the ASR (MS = 0.0) and degraded clean accuracy (CAD = 0.743). With a moderate cost score of 0.5, its Final Score dropped to 0.142. This disparity illustrates the importance of empirical validation: even widely cited defences may underperform in practice. These results confirm that `Safe-DL` can effectively discriminate between defences using consistent, interpretable evaluation metrics.

The *Data Cleaning* defence was applied for data poisoning threats, particularly label-flipping attacks. While its mitigation score was modest (MS = 0.151), it led to an unexpected improvement in clean accuracy (CAD = 1.039) compared to the poisoned baseline. Given its low cost (0.2), this resulted in a Final Score of 0.322. Although the overall mitigation impact was limited, this case demonstrates that `Safe-DL` captures nuanced trade-offs, such as selective improvement in accuracy despite incomplete threat neutralisation, and supports context-aware evaluation of partial defences.

Evasion attacks were tested using two strategies: PGD and SPSA. Against PGD, *Adversarial Training* significantly improved robustness (MS = 0.282) but incurred a clean accuracy drop (CAD = 0.923) and a high cost (0.8), yielding a Final Score of 0.38. *Randomised Smoothing*, in contrast, produced minimal robustness gains (MS = 0.01), moderate degradation (CAD = 0.449), and a low Final Score of 0.093. For SPSA, *Gradient Masking* preserved clean accuracy (CAD = 0.998) but offered negligible robustness (MS = 0.006), leading to a Final Score of 0.196. *JPEG Preprocessing* was the weakest performer overall, with MS = 0.003, CAD = 0.144, and a Final Score of 0.031.

These results show that lightweight or generic defences often fail to generalise against strong adaptive threats. Conversely, techniques like *Activation Clustering* and *Adversarial Training* provide measurable robustness benefits, albeit with trade-offs in cost or accuracy. Most importantly, `Safe-DL` enables a principled, metric-driven comparison of defences using a unified scoring system that incorporates mitigation, utility, and resource cost.

This evaluation validates `Safe-DL` 's capacity to support informed decision-making in adversarial settings. The consistent use of well-defined metrics across all threat types ensures transparency and comparability in defence selection. By exposing the strengths and limitations of each technique, the framework empowers practitioners to make context-appropriate, data-supported choices in securing DL systems.

4.3.3 Usability, Transparency, and Alignment with the *AI Act*

Beyond technical robustness, the value of `Safe-DL` lies in its usability, explainability, and alignment with regulatory expectations such as the *AI Act*. This subsection evaluates these dimensions based on the qualitative criteria established in Section 4.1.3, focusing on ease of configuration, workflow automation, output transparency, and auditability.

From a usability standpoint, `Safe-DL` was designed to support users with limited familiarity with adversarial machine learning. Threat modelling and configuration are initiated through an interactive questionnaire, with responses automatically compiled into a structured YAML profile file that serves as the authoritative configuration for the pipeline. This interface guides users through essential risk dimensions, such as access level, attack goals, and deployment context, without requiring manual coding or internal architectural knowledge. Once defined, the entire pipeline can be executed via a single CLI invocation. Built-in validation mechanisms, intelligent defaults, and human-readable warnings ensure that each module runs with minimal friction, fulfilling the automation and guided workflow support criteria.

Importantly, while the framework provides automation, it does not sacrifice user control. Users can override any attack or defence configuration aspect and inspect intermediate outputs at each step. This enables fine-tuning, debugging, or iterative refinement without disrupting the pipeline's integrity. Each module produces structured outputs in machine-readable formats, such as `.json` and human-readable summaries, like `.md`, which offer a comprehensive trace of all decisions and their rationales. These outputs include raw metrics such as accuracy, attack success rates, and mitigation scores; high-level justifications for selected actions; and tabular or visual representations to support interpretation by both technical and non-technical stakeholders. The `final_report.md` aggregates this information into a coherent, audit-ready document that spans the full security lifecycle from threat modelling to defence scoring.

Regarding regulatory alignment, `Safe-DL` satisfies multiple principles outlined in the *AI Act* for high-risk systems. First, it enables explicit threat documentation and quantification, linking each risk to reproducible simulation results and stored in version-controlled configuration files. Second, all decisions, including defence recommendations, are grounded in prior risk assessments and quantitatively justified. Third, the modular outputs (YAML, JSON, Markdown) promote full reproducibility and transparency, allowing external auditors or regulatory bodies to reconstruct and verify the entire decision process. Finally, the system's modular and abstract design allows easy adaptation to different operational domains, such as healthcare or autonomous vehicles, without requiring fundamental architectural changes.

Together, these features establish that `Safe-DL` not only meets technical robustness standards but also adheres to broader requirements for transparency, auditability, and responsible AI development. The blend of guided configuration, structured outputs, and regulatory foresight positions the framework as a usable and trustworthy tool for practitioners operating in sensitive or high-risk AI contexts.

4.3.4 Comparison with Existing Tools

To contextualise the contributions of `Safe-DL`, it is essential to compare it with other established adversarial machine learning frameworks developed in academia and industry. Among the most prominent tools are IBM's Adversarial Robustness Toolbox (ART), Microsoft's Counterfit, CleverHans, AIX360, and MITRE ATLAS. Each solution focuses on different aspects of the adversarial machine learning lifecycle, from attack generation to explainability and red teaming. However,

they differ significantly in scope, usability, automation, and alignment with audit and compliance needs.

ART stands out as a mature and comprehensive library offering an extensive suite of adversarial attacks and defences. It supports evasion, poisoning, and privacy attacks, such as membership inference and model extraction. It is also compatible with several ML frameworks, including PyTorch, TensorFlow, and Scikit-learn. However, ART primarily targets experienced users. It lacks pipeline-level automation, contextual defence selection, or audit-ready reporting. In contrast, `Safe-DL` sacrifices some breadth in attack coverage in favour of a more structured and guided experience for non-experts.

CleverHans offers high-fidelity implementations of widespread evasion attacks like FGSM, PGD, and C&W. It is widely used in research for reproducibility and benchmarking but is very low-level and lacks defences, pipeline orchestration, or reporting support. While CleverHans outperforms `Safe-DL` regarding attack breadth and customisation, it does not aim to support practical or audit-ready workflows.

Counterfit provides a CLI for performing adversarial testing in black-box scenarios. Integrating existing libraries like ART allows users to test deployed APIs or remote models for vulnerability. Its strength lies in penetration testing for real-world systems. However, it lacks support for defence mechanisms, defence evaluation, or integration of contextual risk modelling. Compared to `Safe-DL`, Counterfit is more suitable for adversarial red-teaming than lifecycle analysis or mitigation planning.

AIX360 is an interpretability-focused toolkit that enables users to apply post-hoc explainability techniques such as SHAP, LIME, and Anchors. While it improves transparency in AI decision-making, it does not deal with robustness or adversarial threats. Therefore, AIX360 and `Safe-DL` occupy complementary spaces: one focused on explainability, the other on security.

MITRE ATLAS provides a curated taxonomy of adversarial tactics, known threat scenarios, and mitigation techniques. It serves as a strategic guide and knowledge base but is not an executable toolkit. Unlike `Safe-DL`, it does not include attack implementations, defences, or evaluation pipelines. Nonetheless, ATLAS can inform the design of systems like `Safe-DL` by framing real-world threat models and remediation strategies.

In comparison, `Safe-DL` delivers a structured, modular pipeline that spans threat modelling, attack simulation, risk assessment, automated defence selection, and final reporting. It emphasises user-friendliness through YAML configuration, interpretable outputs in Markdown and JSON, and decision guidance through scoring metrics. While its current version supports fewer threat types than ART or Counterfit, its unique contribution lies in integrating the full adversarial security lifecycle into a cohesive and auditable workflow.

It is important to clarify that the comparison presented here is feature-based and not empirical. That is, the evaluation of existing frameworks was conducted by analysing their official documentation, published literature, and available source code rather than executing them under the same experimental conditions used for `Safe-DL`. While a hands-on benchmark across identical scenarios would provide further validation, this was beyond the scope of the current work.

The comparison, therefore, focuses on supported features, usability, automation capabilities, and reporting mechanisms.

Table 4.6 summarises key features across all frameworks discussed.

Table 4.6: Feature comparison between `Safe-DL` and related adversarial ML frameworks.

Framework	Threats	Pipeline	Defence Eval	Usability	Reporting	Extensible
`Safe-DL`	✓	✓	✓	✓	✓	✓
ART (IBM)	✓	✗	△	✗	△	✓
CleverHans	✓	✗	✗	✗	✗	△
Counterfit	✓	△	✗	△	△	△
AIX360	✗	✗	✗	✓	✓	✓
MITRE ATLAS	△	✗	✗	△	✓	△

Legend: ✓= Fully supported; △= Partially supported; ✗= Not supported.

Despite its strengths, `Safe-DL` still has limitations. Although privacy attacks such as model inversion, membership inference, and model extraction were considered during the design phase, they remain unimplemented in the current version. These attack types are increasingly relevant for real-world deployments and regulatory compliance, and their inclusion is a key direction for future work. Existing frameworks such as ART already support some of these techniques and could be integrated to extend `Safe-DL` 's capabilities.

While no tool addresses all aspects of adversarial machine learning, `Safe-DL` fills a specific gap by combining lifecycle integration, user accessibility, and audit readiness in a single cohesive platform. Its contribution is not to replace low-level libraries or expert tools but to bridge them into a workflow suitable for secure deployment and regulatory compliance.

4.3.5 Limitations of the Current Prototype

While `Safe-DL` demonstrates the feasibility of a modular, auditable, and user-guided security framework for DL, it has limitations. These constraints, identified during development and evaluation, point to several avenues for future improvement.

First, the current prototype offers incomplete threat coverage. Although privacy attacks such as model extraction, membership inference, and model inversion were considered during the design phase, they were not implemented. These attacks are increasingly relevant for regulatory compliance and real-world deployment scenarios. Moreover, while `Safe-DL` supports evasion, poisoning, and backdoor attacks, the number of implemented techniques per category remains limited. Larger-scale libraries such as ART provide more exhaustive coverage in this regard.

Physical-world threats, such as adversarial patches, printed perturbations, or sensor interference, are also outside the scope of the current framework. The system assumes purely digital input pipelines, which limits its applicability to specific deployment contexts like autonomous vehicles or edge computing devices.

Despite its modularity, extending the framework with new attacks or defences requires familiarity with its internal architecture. This includes understanding the directory structure, base classes, and module dependencies. For users without a development background, this poses a barrier to customisation. Similarly, the absence of a graphical user interface makes the system less accessible for broader adoption, especially in enterprise or educational settings where visual workflows are preferred.

Another area of concern is the modelling of defence costs. The Defence Cost Score used in Module 5 is based on manually estimated levels rather than empirical runtime metrics such as training time, memory usage, or Floating-point Operations Per Second (FLOPs). This introduces a subjective element that could be improved with automated profiling in future versions. Relatedly, the current outputs are limited to text-based formats. While the Markdown and JSON files support traceability and reproducibility, they do not offer interactive visualisation or real-time monitoring, which would benefit production environments.

The framework is also currently restricted to supervised image classification tasks. It was evaluated on datasets such as CIFAR-10 and MNIST and does not generalise directly to tasks like object detection, regression, or natural language processing. Furthermore, `Safe-DL` is intended for static evaluation and is not integrated into continuous deployment workflows, such as those involving Machine Learning Operations (MLOps) platforms, model registries, or CI/CD pipelines.

From a functionality perspective, the framework evaluates one defence at a time. There is no native support for assessing defence combinations, such as adversarial training followed by input preprocessing. While parameter values for attacks and defences are often inferred or recommended automatically, optimal tuning still requires manual intervention for deployment-specific constraints.

Results obtained through standard academic datasets may also fail to generalise to real-world industrial applications, particularly those involving large-scale proprietary models and highly imbalanced data distributions. Additionally, by covering the full security lifecycle, `Safe-DL` faces the risk of overextension. While the current implementation performs consistently across modules, ongoing development must ensure that future features maintain coherence and do not dilute overall robustness or usability.

A further limitation concerns the comparative evaluation. Although Section 4.3.4 outlines a detailed feature-level analysis of related frameworks, these tools were not empirically tested on the same tasks used for `Safe-DL`. The comparison was based solely on documentation, public APIs, and a literature review. A more rigorous and fair benchmarking across identical threat scenarios would offer more profound insights into relative effectiveness and remains a priority for future work.

Although these limitations do not undermine the utility or vision of `Safe-DL`, they highlight concrete opportunities for refinement. Addressing them will improve the framework's usability, generalisation, and trustworthiness in practical adversarial machine learning deployments.

4.3.6 Engineering and Development Challenges

The development of `Safe-DL` presented various engineering challenges stemming from its modular architecture, diverse functionality, and design emphasis on usability and auditability. This subsection reflects on the most critical technical and design obstacles encountered during implementation.

One major challenge was the integration of heterogeneous modules. Each component in the pipeline targets a different phase of the security lifecycle, from threat modelling to final reporting, requiring consistent communication between modules. To ensure modularity without sacrificing interoperability, a file-based communication interface was adopted using structured formats such as YAML, JSON, and Markdown. This approach demanded careful data schemas and directory structure planning to minimise coupling while ensuring robustness.

Designing an architecture that was both modular and extensible also introduced early complexity. Abstract base classes and shared configuration utilities were implemented to allow for future integration of new attacks, defences, or evaluation mechanisms. This extensibility had to be balanced against maintainability, and special attention was paid to ensuring consistency in how modules interact, configure themselves, and log results.

Balancing automation with user control emerged as a key goal. The configuration system, built around a structured YAML profile, needed to support default values, semantic validation, and selective overrides. This required a flexible yet reliable parsing layer to provide meaningful feedback and prevent silent failures. The CLI interface was designed to abstract internal complexity while allowing power users to retain flexibility. Interactive guidance, intelligent defaults, and inline help messages were added to streamline user experience and minimise misconfiguration.

Another important challenge involved metric normalisation and report consistency. The framework produces diverse metrics across modules, including Attack Success Rate, Risk Score, Mitigation Score, and Clean Accuracy Drop. Normalisation strategies were required to ensure these outputs were interpretable and comparable within a unified evaluation process. These metrics were harmonised through shared schemas, allowing them to be aggregated into final evaluation scores and presented in consistent formats.

Auditability and artifact traceability were central design goals. Structured logging and consistent output generation were enforced throughout the pipeline, with Markdown summaries and machine-readable reports created at each stage. To prevent audit gaps, consistency checks were implemented to catch missing files or misaligned metrics before final reporting.

The computational demands of certain defences and attacks posed further difficulties. Techniques like adversarial training and gradient-based attacks were time- and resource-intensive, slowing down development and testing. Caching mechanisms and testing on reduced datasets were introduced to mitigate this and enable faster iteration cycles during development.

Supporting generalisation across datasets and models introduced additional complexity. Since users may choose arbitrary datasets or bring their own models, all modules had to operate through abstracted interfaces. This required flexible input pipelines, model wrappers, and normalisation

layers that could adapt to diverse scenarios without breaking consistency. The system was validated across dataset benchmarks, including CIFAR-10 and MNIST with CNN and Resnet18, to ensure reliability under multiple deployment contexts.

Together, these engineering challenges highlight the complexity of designing a real-world adversarial robustness pipeline that is both rigorous and user-friendly. The solutions developed to address them form a solid foundation for future extensions and reflect the system's capacity to evolve while maintaining a high standard of usability and traceability.

Taken together, the analyses presented in this section confirm that `Safe-DL` fulfils its core objectives by offering an integrated, transparent, and practical approach to adversarial robustness. The framework successfully combines threat modelling, automated defence recommendation, and structured evaluation into a coherent pipeline accessible to non-experts yet rigorous enough for high-stakes contexts. Its quantitative performance across diverse threat types, alignment with regulatory principles, and emphasis on usability position it as a distinctive contribution within the adversarial machine learning landscape. At the same time, a realistic assessment of current limitations and development challenges reveals clear directions for future refinement, including expanding threat coverage, improving extensibility, and strengthening empirical comparisons with other tools. These findings underscore the potential and the maturity of `Safe-DL` as a foundation for secure, auditable, and regulation-aware AI systems.

4.4 Chapter Summary

This chapter comprehensively validated the `Safe-DL` framework through a realistic end-to-end scenario. It began by detailing the experimental setup, including a simulated industrial use case, dataset and model choices, and evaluation criteria. This foundation enabled a structured walkthrough of the entire adversarial security lifecycle as implemented by `Safe-DL`.

The execution phase highlighted how each of the six modules contributes to the pipeline: from translating threat profiles into attack configurations to simulating adversarial behaviour, quantifying risk, applying defences, evaluating mitigation outcomes, and generating audit-ready reports. Key design principles such as modularity, auditability, and automation were demonstrated throughout this process.

The final part of the chapter engaged in critical analysis, assessing the framework's alignment with its original goals, effectiveness across threat categories, and relevance in regulatory and practical contexts. It also included a comparison with related tools, an enumeration of current limitations, and reflections on the engineering challenges faced during development.

These elements confirm that `Safe-DL` is a technically sound and practically usable system for adversarial robustness evaluation. Its structured design and extensibility allow it to serve as a foundation for further research, tool development, or industrial deployment.

Chapter 5 builds on these insights to consolidate the dissertation's contributions and outline concrete directions for future development and broader adoption.

Chapter 5

Conclusion and Future Work

This final chapter provides a consolidated overview of the project, summarising its main contributions, technical insights, and opportunities for future development. Rather than revisiting specific results, the focus here is on extracting broader lessons, assessing the framework's long-term relevance, and outlining actionable next steps.

The chapter begins with synthesising the scientific and engineering contributions delivered by `Safe-DL` in terms of methodology and implementation. It then offers critical reflections on the framework's design decisions, practical applicability, and alignment with the needs of non-expert users and industry stakeholders. The chapter concludes with a roadmap for future work, identifying concrete improvements and directions to expand the framework's scope, usability, and deployment readiness.

5.1 Summary of Contributions

This section presents a consolidated overview of the contributions introduced by the `Safe-DL` framework in scientific methodology and practical engineering. These contributions reflect the broader goal of advancing secure DL through tools that are not only technically rigorous but also transparent, auditable, and usable in applied contexts.

On the conceptual side, `Safe-DL` proposes a structured approach to adversarial robustness that emphasises lifecycle thinking and reproducibility. On the engineering side, it delivers a working system that implements this vision through modular components, unified configuration, and traceable outputs, all tested and validated in diverse evaluation settings.

5.1.1 Scientific and Methodological Contributions

`Safe-DL` introduces several scientific and methodological innovations that advance the practice of secure DL. Rather than limiting itself to implementing specific attacks or defences, the framework presents a systematic, reproducible, and auditable pipeline that serves research and practical security needs.

One of its core methodological contributions is formally structuring the secure development lifecycle for DL into six well-defined modules, covering stages from threat modelling to final reporting. This decomposition provides a conceptual blueprint for organising adversarial evaluations coherently and modularly, enabling each stage to be developed, extended, or audited independently.

Another key contribution is adopting a unified configuration mechanism via a central YAML profile file. This file progressively accumulates contextual information, parameters, and outputs across the pipeline, facilitating traceability, reproducibility, and human-in-the-loop workflows.

`Safe-DL` also operationalises compound evaluation metrics for comparing defence strategies in a principled way. For example, the `final_score` aggregates mitigation effectiveness, clean accuracy preservation, and estimated defence cost into a single composite metric. This addresses a common shortcoming in existing tools, which often report raw metrics in isolation and lack mechanisms for trade-off analysis.

Auditability and explainability are integral to the framework's design. All key outputs, including decisions, metrics, and justifications, are captured in machine-readable formats (JSON) and human-readable reports (Markdown). These artifacts can communicate results across technical and non-technical audiences, support formal documentation, and assist in regulatory compliance.

By embedding these principles into a coherent and practical system, `Safe-DL` helps bridge the gap between research prototypes and usable security workflows. Its methodological foundations support secure AI development in environments that demand traceability, extensibility, and alignment with real-world constraints.

5.1.2 Functional and Engineering Deliverables

From an engineering standpoint, `Safe-DL` was developed as a modular, extensible, and fully functional framework for evaluating the security of DL systems. It moves beyond the scope of a research prototype by delivering a cohesive and reusable software platform capable of supporting end-to-end adversarial risk assessments. Its architecture was principled and practical, enabling real-world usage while maintaining consistency with sound engineering practices.

At the core of the system are six well-defined modules, each corresponding to a phase in the adversarial security lifecycle:

- *Module 1 – Threat Modelling:* interprets a user-defined risk profile and maps it to concrete attack strategies.
- *Module 2 – Attack Simulation:* launches automated adversarial evaluations based on the threat model.
- *Module 3 – Risk Assessment:* quantifies threat impact using structured metrics and prioritization scores.
- *Module 4 – Defence Application:* selects and applies relevant defence techniques.

- *Module 5 – Defence Evaluation:* ranks defence strategies based on effectiveness, clean accuracy drop, and cost.
- *Module 6 – Final Reporting:* generates an interpretable Markdown summary and a structured JSON trace.

All decisions and configurations across the pipeline are managed through the central YAML profile file, which acts as an input specification and a persistent record of the system's execution state. This unified configuration enables traceability, reproducibility, and seamless context propagation between modules.

The framework also provides an interactive CLI built with the `questionary` library, allowing users to configure and operate the system through guided prompts. This design makes the framework accessible to non-experts while preserving complete control for advanced users.

The framework produces outputs in structured formats suitable for both interpretability and automation. These comprise Markdown reports with evaluation summaries, machine-readable JSON traces, and image directories containing adversarial inputs or training instances filtered out by selected defences.

Extensibility is supported through clear directory conventions and consistent function interfaces between modules. New components, such as attacks, defences, datasets, or model types, can be added as long as they respect the expected inputs and outputs of adjacent stages in the pipeline. While no formal APIs are defined, the internal logic follows a modular structure that enables straightforward integration for developers familiar with the codebase.

Thanks to these features, `Safe-DL` is a reliable and extensible framework enabling principled security evaluations and supporting real-world deployment needs.

Together, these scientific and engineering contributions demonstrate that `Safe-DL` is a theoretically grounded framework and a practical tool for adversarial robustness assessment. Combining methodological clarity with usable software bridges a gap between academic research and real-world AI security practices. These contributions set the stage for the broader discussion in Section 5.2, which critically reflects the project's development, relevance, and practical implications.

5.2 Final Reflections

This section offers a broader reflection on the development and significance of the `Safe-DL` framework. The focus is not on individual components but on the overall design trajectory, guiding principles, and the framework's relevance in practical and regulatory contexts.

It seeks to distil the key insights that emerged throughout the project, considering both conceptual and implementation perspectives. Doing so helps clarify the role that a framework like `Safe-DL` can play in enabling more secure, transparent, and responsible AI systems.

5.2.1 Lessons Learned During Development

The development of `Safe-DL` provided important lessons across software engineering, usability design, and adversarial robustness methodology. These insights reflect the challenges encountered during implementation and the broader design decisions that shaped the project.

One of the key takeaways was the importance of disciplined modularity. Creating a modular system required more than simply separating code into files. It also demanded well-defined interfaces, consistent data structures, and loose coupling between components. The early investment in defining shared schemas and predictable data flows was essential for ensuring maintainability and extensibility.

Using a central configuration file also proved to be a strategic design decision. It allowed each module to operate independently while preserving global context throughout the pipeline. This enabled traceability and simplified debugging, although it also introduced incremental validation and consistency management challenges.

Balancing automation with user control emerged as another recurring challenge. While the framework provides intelligent defaults and guided flows, it was necessary to ensure that expert users could override parameters without breaking the pipeline's internal logic. This led to the development of robust parsing mechanisms and explicit error messaging to support flexible yet safe configurations.

Computational cost was also a practical constraint. Techniques such as adversarial training or iterative optimisation required significant processing time, which slowed testing and development. Caching strategies and reduced test sets were used to mitigate this and accelerate feedback loops during iterative implementation.

The need to normalise metrics became apparent when consolidating results across modules. Each stage of the pipeline generated outputs with different formats and interpretations, and without a unifying schema, the final reports would have lacked coherence. Aligning metrics such as ASR, clean accuracy, and mitigation effectiveness allowed for more interpretable comparisons and trade-off analysis.

Auditability evolved into a central design principle. Although not a primary concern in the early stages, it became clear that traceable outputs such as YAML traces, Markdown summaries, and structured JSON logs would be critical for regulatory alignment and reproducibility. This shift influenced both the structure of the pipeline and the reporting strategy.

Building a user-friendly CLI interface also brought important lessons. The goal was to support both novices and advanced users without compromising functionality. Tools like `questionary` helped implement guided prompts, but maintaining internal state, managing conditional flows, and handling exceptions added significant complexity to the implementation.

Finally, scope management played a crucial role. The initial vision included broad coverage across attack types, defences, and datasets, but attempting to implement everything simultaneously risked overextension. Prioritising core functionality and delivering a minimal viable product proved to be the right approach for ensuring coherence and robustness.

These lessons emphasise that building secure, usable AI tools requires much more than technical correctness. It also demands careful architectural planning, a focus on usability, and a willingness to iterate and refine based on evolving constraints and insights.

5.2.2 Relevance of Secure Deep Learning Today

As machine learning systems increasingly inform decisions in critical domains such as healthcare, finance, transportation, and public administration, the robustness of DL models has become a foundational requirement. The risks associated with adversarial manipulation are well documented: models can be deceived, corrupted, or subverted through carefully crafted perturbations that often remain imperceptible to humans.

Despite this, security remains under-prioritised in most production ML pipelines, which continue to emphasise performance and scalability. Adversarial machine learning is still underrepresented in industry-grade tooling, particularly outside academic and high-assurance contexts. `Safe-DL` addresses this gap by providing a modular pipeline that integrates recent advances in adversarial research with practical, auditable workflows.

The importance of secure DL is further underscored by evolving legal and ethical standards. In particular, the *AI Act* mandates that high-risk AI systems comply with principles such as robustness, transparency, traceability, and risk mitigation. These are not just best practices but binding regulatory requirements.

In this landscape, `Safe-DL` takes a concrete step toward operationalising security by design. Its structured approach to threat modelling, attack simulation, and defence evaluation equips practitioners to assess robustness through explainable and repeatable methods. The emphasis on audit-ready outputs aligns with industry trends in explainable and trustworthy AI.

Secure DL is, therefore, increasingly viewed as a prerequisite for responsible AI. Tools like `Safe-DL` can help democratise access to rigorous robustness evaluations and promote a development culture rooted in safety, accountability, and technical integrity.

5.2.3 Usefulness for Non-experts and Industry

One of the core motivations behind the design of `Safe-DL` was to make adversarial robustness workflows accessible to practitioners who may not have deep expertise in security or adversarial machine learning. While existing tools are often powerful, they tend to assume familiarity with attack categories, tuning strategies, and low-level implementation details, which limits their applicability in typical engineering workflows.

`Safe-DL` addresses this gap through a combination of guided configuration, sensible defaults, and a modular CLI. A central YAML configuration file drives the entire security pipeline, which users can customise incrementally. Only the parameters of interest need to be defined, with the framework providing intelligent defaults and consistency checks to handle the rest.

Each module also produces outputs that support both human understanding and formal documentation. These include Markdown summaries, JSON traces, and visual artifacts such as adversarial examples or attack statistics. The resulting audit report is intended to be interpretable by engineers and compliance teams, auditors, and other stakeholders responsible for oversight or certification.

The modularity of the system enables partial adoption. Organisations may run only selected phases, such as threat modelling or attack simulation, and integrate those results into existing CI/CD workflows or internal dashboards. This flexibility broadens the framework's relevance across use cases ranging from academic prototyping to industry-grade assurance.

By lowering the technical barrier to security evaluation and producing both interpretable and auditable outputs, `Safe-DL` helps bridge the gap between cutting-edge adversarial ML and the operational realities of engineering teams, risk managers, and regulatory actors.

These reflections emphasise the complexity of building secure, usable, and regulatory-aligned DL systems. Throughout the development of `Safe-DL`, attention to modularity, usability, and auditability proved essential to making adversarial robustness practical for diverse users. As the importance of secure AI continues to grow, aligning technical design with broader deployment and compliance needs becomes increasingly critical. Section 5.3 outlines concrete directions for extending the framework's scope, improving its capabilities, and deepening its real-world impact.

5.3 Future Work

While `Safe-DL` already provides a modular and usable framework for adversarial risk assessment, several areas remain open for further development. Some of these stem from implementation trade-offs made during the initial design phase, while others reflect broader gaps in the current landscape of adversarial robustness tooling.

This section outlines concrete opportunities to strengthen and expand the framework. These include technical enhancements to improve performance and usability, integrating new threat models and defence strategies, and broader evaluation efforts that test `Safe-DL` in enterprise settings and regulated domains. Together, these directions chart a path for transforming `Safe-DL` into a more comprehensive and widely deployable system for secure machine learning.

5.3.1 Technical Improvements and Extensions

Although `Safe-DL` is already functional and usable, several technical enhancements could significantly improve its robustness, scalability, and overall user experience. One important direction involves supporting multi-defence pipelines. Currently, only one defence can be applied per evaluation cycle, but in practice, combining multiple defences, such as adversarial training followed by input sanitisation, often leads to improved robustness. Enabling sequential or parallel composition of defences, along with metrics to assess their cumulative impact, would make the framework more versatile.

Another area of improvement concerns real-time monitoring and visualisation. While the framework already generates structured reports in Markdown and JSON, these outputs are static and not interactive. A dedicated dashboard module, implemented with tools like Streamlit[1] or Plotly[2], could allow users to inspect attack dynamics, defence effectiveness, and overall system behaviour in a more intuitive and accessible way.

The modelling of defence costs also presents room for refinement. At present, cost scores are manually defined. A more rigorous approach would involve logging runtime statistics such as training time, memory usage, or FLOPs, providing empirical support for trade-off analysis and making the DCS more informative.

Enhancements to the CLI could also increase usability. Context-aware help messages, more thoughtful default recommendations, and validation of configuration options before execution would help users avoid errors and streamline their interaction with the system.

The central YAML configuration file could benefit from automated schema validation. Currently, field validation is handled manually, but tools like Pydantic[3] or Cerberus[4] could enforce structure and type correctness systematically, reducing errors and improving feedback during configuration.

Parallel and asynchronous execution support is another promising area. Several pipeline stages, including the execution of multiple attacks or the evaluation of different defences, exhibit inherent parallelism and can be executed concurrently without interdependencies. Multiprocessing or GPU-aware scheduling could significantly reduce evaluation time, especially in resource-rich environments.

Broader integration with MLOps ecosystems would also enhance adoption in enterprise contexts. Connecting `Safe-DL` with platforms like MLflow[5] or Airflow[6], or exposing its functionality through a REST API, would enable continuous evaluation and tighter integration with production workflows.

Taken as a whole, such improvements would significantly strengthen `Safe-DL` 's core functionality while paving the way for broader adoption across a range of development and deployment environments.

5.3.2 Expanded Attack and Defence Support

A key strength of `Safe-DL` lies in its modular design, which allows for gradually incorporating new threat categories and mitigation strategies. Future work should utilise this flexibility to expand the diversity of supported attacks and the sophistication of available defences.

One important direction is the inclusion of privacy-focused attacks. The current implementation does not yet support threats such as membership inference, model extraction, or model

[1] See: https://streamlit.io
[2] See: https://plotly.com
[3] See: https://docs.pydantic.dev
[4] See: https://docs.python-cerberus.org
[5] See: https://mlflow.org
[6] See: https://airflow.apache.org

inversion, which compromise model confidentiality rather than robustness. Adding these capabilities would align the framework more closely with regulations like the GDPR and the *AI Act* while encouraging defences such as differential privacy and model watermarking.

Another valuable extension is the support for combined defence strategies. Currently, `Safe-DL` applies only one defence per evaluation cycle. Enabling the sequential or parallel application of multiple defences, such as combining adversarial training with input sanitisation, would allow for exploring synergies between techniques. This would require architectural changes to accommodate composite execution and cumulative evaluation.

Improving the modelling of defence costs is also a relevant goal. While current estimates are assigned manually, future versions could log runtime metrics such as training duration, memory consumption, and FLOPs. These empirical indicators enable more accurate defence comparisons and better inform trade-off decisions.

Broadening support beyond image classification would increase the framework's applicability across domains. Adapting to modalities like text, tabular data, or audio would involve generalising key components, such as data loaders and normalisation logic. These adaptations could be integrated with moderate development effort thanks to the modular separation.

By extending its coverage to a broader array of threat types and data domains, `Safe-DL` has the potential to become a more comprehensive and versatile platform for adversarial security across real-world machine learning applications.

5.3.3 Broader Evaluation and Deployment

For `Safe-DL` to achieve its full potential as a practical tool for adversarial robustness, it must be validated and adopted across diverse real-world environments. So far, the framework has been tested primarily in academic scenarios with standard datasets and controlled setups. Broader evaluation would help uncover usability gaps, integration constraints, and corner cases not captured in isolated experiments, reinforcing the framework's reliability and relevance.

A more rigorous empirical comparison with existing adversarial ML frameworks is a natural next step. Although a feature-level analysis was presented in Section 4.3.4, those tools were not evaluated under the same scenarios as `Safe-DL`. Future versions should include benchmark experiments across identical datasets, threat models, and evaluation metrics to ensure a fair and transparent comparison of effectiveness, usability, and extensibility.

Beyond technical benchmarking, validation in regulated environments is a key opportunity. With increasing pressure from policies such as the *AI Act*, there is a growing demand for tools that support explainability, traceability, and robustness. `Safe-DL` could contribute to regulatory compliance processes by generating standardised, human-readable reports that document the security posture of AI systems in a structured and verifiable manner.

Engaging practitioners in user studies would provide valuable feedback on the framework's accessibility and learning curve. Collaborations with industry users or non-expert testers could uncover friction points and inspire new usability features, including GUI interfaces, wise configuration wizards, or recommendation systems for parameter tuning.

Expanding evaluation across operational, regulatory, and usability contexts will be essential for transforming `Safe-DL` from a research prototype into a dependable component of modern AI assurance ecosystems.

The directions outlined in this section reflect both immediate priorities and long-term opportunities for the evolution of `Safe-DL`. Enhancing technical robustness, broadening the scope of supported threats and mitigations, and validating the framework in operational and regulatory settings are critical steps toward consolidating its value. These efforts will help ensure that `Safe-DL` remains a conceptually sound tool for adversarial robustness and a practical and trustworthy solution for real-world AI development and assurance.

5.4 Chapter Summary

This chapter presented a comprehensive reflection on the contributions, challenges, and future directions of the `Safe-DL` framework. It summarised the scientific and engineering outcomes, highlighting the framework's dual role as a methodological proposal and a functional tool for evaluating adversarial robustness.

The following reflections examined key insights gained during development, reinforced the importance of secure DL in contemporary AI practice, and discussed the framework's potential to support non-expert users and adapt to industrial workflows. Each aspect contributes to the broader goal of making adversarial evaluation more accessible, transparent, and aligned with practical needs.

A final section outlined concrete directions for future work, focusing on architectural improvements, expanded threat coverage, and validation in real-world and regulated environments. These efforts aim to consolidate `Safe-DL` as a usable, extensible, and regulation-aware platform for secure AI development.

References

AKHTAR, Naveed; MIAN, Ajmal; KARDAN, Navid; SHAH, Mubarak, 2021. Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey. *IEEE Access* [online]. Vol. 9, pp. 155161–155196 [visited on 2024-11-02]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2021.3127960.

AL-ANDOLI, Mohammed Nasser; TAN, Shing Chiang; SIM, Kok Swee; GOH, Pey Yun; LIM, Chee Peng, 2024. A Framework for Robust Deep Learning Models Against Adversarial Attacks Based on a Protection Layer Approach. *IEEE Access* [online]. Vol. 12, pp. 17522–17540 [visited on 2024-12-19]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2024.3354699. Conference Name: IEEE Access.

ARSHAD, Iram; ALSAMHI, Saeed Hamood; QIAO, Yuansong; LEE, Brian; YE, Yuhang, 2023. A Novel Framework for Smart Cyber Defence: A Deep-Dive Into Deep Learning Attacks and Defences. *IEEE Access* [online]. Vol. 11, pp. 88527–88548 [visited on 2025-04-16]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2023.3306333.

BADJIE, Bakary; CECÍLIO, José; CASIMIRO, Antonio, 2024. Adversarial Attacks and Countermeasures on Image Classification-based Deep Learning Models in Autonomous Driving Systems: A Systematic Review. *ACM Comput. Surv.* [online]. Vol. 57, no. 1, 20:1–20:52 [visited on 2025-06-16]. ISSN 0360-0300. Available from DOI: 10.1145/3691625.

BARACALDO, Nathalie; CHEN, Bryant; LUDWIG, Heiko; SAFAVI, Jaehoon Amir, 2017. Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* [online]. New York, NY, USA: Association for Computing Machinery, pp. 103–110 [visited on 2025-06-18]. AISec '17. ISBN 978-1-4503-5202-4. Available from DOI: 10.1145/3128572.3140450.

BRENDEL, Wieland; RAUBER, Jonas; BETHGE, Matthias, 2018. *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models* [online]. arXiv [visited on 2025-06-18]. No. arXiv:1712.04248. Available from DOI: 10.48550/arXiv.1712.04248.

CHEN, Hongsong; ZHANG, Yongpeng; CAO, Yongrui; XIE, Jing, 2021. Security issues and defensive approaches in deep learning frameworks. *Tsinghua Science and Technology* [online]. Vol. 26, no. 6, pp. 894–905 [visited on 2024-11-02]. ISSN 1007-0214. Available from DOI: 10.26599/TST.2020.9010050.

COSTA, Joana C.; ROXO, Tiago; PROENÇA, Hugo; INÁCIO, Pedro Ricardo Morais, 2024. How Deep Learning Sees the World: A Survey on Adversarial Attacks & Defenses. *IEEE Access* [online]. Vol. 12, pp. 61113–61136 [visited on 2024-12-19]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2024.3395118. Conference Name: IEEE Access.

DANG, Tran Khanh; TRUONG, Phat T. Tran; TRAN, Pi To, 2020. Data Poisoning Attack on Deep Neural Network and Some Defense Methods. In: *2020 International Conference on Advanced Computing and Applications (ACOMP)* [online], pp. 15–22 [visited on 2024-11-02]. Available from DOI: 10.1109/ACOMP50827.2020.00010.

DENG, Yao; ZHENG, Xi; ZHANG, Tianyi; CHEN, Chen; LOU, Guannan; KIM, Miryung, 2020. An Analysis of Adversarial Attacks and Defenses on Autonomous Driving Models. In: *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)* [online], pp. 1–10 [visited on 2025-06-16]. Available from DOI: 10.1109/PerCom45495.2020.9127389. ISSN: 2474-249X.

FINLAYSON, Samuel G.; BOWERS, John D.; ITO, Joichi; ZITTRAIN, Jonathan L.; BEAM, Andrew L.; KOHANE, Isaac S., 2019. Adversarial attacks on medical machine learning. *Science* [online]. Vol. 363, no. 6433, pp. 1287–1289 [visited on 2025-06-16]. ISSN 0036-8075, ISSN 1095-9203. Available from DOI: 10.1126/science.aaw4399.

GAO, Yansong; DOAN, Bao Gia; ZHANG, Zhi; MA, Siqi; ZHANG, Jiliang; FU, Anmin; NEPAL, S.; KIM, Hyoungshick, 2020. Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review. *ArXiv* [online] [visited on 2025-04-16]. Available from: https://www.semanticscholar.org/paper/69dd1b9e8391430a667214a9ca6c0bc94560deb2.

GAO, Yansong; XU, Chang; WANG, Derui; CHEN, Shiping; RANASINGHE, Damith C.; NEPAL, Surya, 2020. *STRIP: A Defence Against Trojan Attacks on Deep Neural Networks* [online]. arXiv [visited on 2025-06-18]. No. arXiv:1902.06531. Available from DOI: 10.48550/arXiv.1902.06531.

GIRDHAR, Mansi; HONG, Junho; MOORE, John, 2023. Cybersecurity of Autonomous Vehicles: A Systematic Literature Review of Adversarial Attacks and Defense Models. *IEEE Open Journal of Vehicular Technology* [online]. Vol. 4, pp. 417–437 [visited on 2025-06-16]. ISSN 2644-1330. Available from DOI: 10.1109/OJVT.2023.3265363.

GONG, Xueluan; WANG, Ziyao; LI, Shuaike; CHEN, Yanjiao; WANG, Qian, 2023. A GAN-Based Defense Framework Against Model Inversion Attacks. *IEEE Transactions on Information Forensics and Security* [online]. Vol. 18, pp. 4475–4487 [visited on 2025-06-18]. ISSN 1556-6021. Available from DOI: 10.1109/TIFS.2023.3295944.

GUESMI, Amira; ALOUANI, Ihsen; BAKLOUTI, Mouna; FRIKHA, Tarek; ABID, Mohamed, 2022. SIT: Stochastic Input Transformation to Defend Against Adversarial Attacks on Deep Neural Networks. *IEEE Design & Test* [online]. Vol. 39, no. 3, pp. 63–72 [visited on 2024-12-19]. ISSN 2168-2364. Available from DOI: 10.1109/MDAT.2021.3077542. Conference Name: IEEE Design & Test.

GUO, Jun; BAO, Wei; WANG, Jiakai; MA, Yuqing; GAO, Xinghai; XIAO, Gang; LIU, Aishan; DONG, Jian; LIU, Xianglong; WU, Wenjun, 2023. A comprehensive evaluation framework for deep model robustness. *Pattern Recognition* [online]. Vol. 137, p. 109308 [visited on 2025-04-16]. ISSN 0031-3203. Available from DOI: 10.1016/j.patcog.2023.109308.

GUO, Wei; TONDI, Benedetta; BARNI, Mauro, 2022. An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences. *IEEE Open Journal of Signal Processing* [online]. Vol. 3, pp. 261–287 [visited on 2024-11-03]. ISSN 2644-1322. Available from DOI: 10.1109/OJSP.2022.3190213.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian, 2015. *Deep Residual Learning for Image Recognition* [online]. arXiv [visited on 2025-06-22]. No. arXiv:1512.03385. Available from DOI: 10.48550/arXiv.1512.03385.

HU, Charles; HU, Yen-Hung Frank, 2020. Data Poisoning on Deep Learning Models. In: *2020 International Conference on Computational Science and Computational Intelligence (CSCI)* [online], pp. 628–632 [visited on 2024-11-03]. Available from DOI: 10.1109/CSCI51800.2020.00111.

HU, Hongsheng; SALCIC, Zoran; SUN, Lichao; DOBBIE, Gillian; YU, Philip S.; ZHANG, Xuyun, 2022. Membership Inference Attacks on Machine Learning: A Survey. *ACM Computing Surveys* [online]. Vol. 54, no. 11, pp. 1–37 [visited on 2025-06-18]. ISSN 0360-0300, ISSN 1557-7341. Available from DOI: 10.1145/3523273.

HU, Li; YAN, Anli; YAN, Hongyang; LI, Jin; HUANG, Teng; ZHANG, Yingying; DONG, Changyu; YANG, Chunsheng, 2023. Defenses to Membership Inference Attacks: A Survey. *ACM Comput. Surv.* [online]. Vol. 56, no. 4, 92:1–92:34 [visited on 2025-06-18]. ISSN 0360-0300. Available from DOI: 10.1145/3620667.

HUANG, Hai; MU, Jiaming; GONG, Neil Zhenqiang; LI, Qi; LIU, Bin; XU, Mingwei, 2021. Data Poisoning Attacks to Deep Learning Based Recommender Systems. *Proceedings 2021 Network and Distributed System Security Symposium* [online] [visited on 2025-06-18]. Available from DOI: 10.14722/ndss.2021.24525. Conference Name: Network and Distributed System Security Symposium ISBN: 9781891562662 Place: Virtual Publisher: Internet Society.

JIANG, Wenbo; LI, Hongwei; XU, Guowen; ZHANG, Tianwei; LU, Rongxing, 2024. A Comprehensive Defense Framework Against Model Extraction Attacks. *IEEE Transactions on Dependable and Secure Computing* [online]. Vol. 21, no. 2, pp. 685–700 [visited on 2024-11-04]. ISSN 1941-0018. Available from DOI: 10.1109/TDSC.2023.3261327.

JUUTI, Mika; SZYLLER, Sebastian; MARCHAL, Samuel; ASOKAN, N., 2019. *PRADA: Protecting against DNN Model Stealing Attacks* [online]. arXiv [visited on 2025-06-18]. No. arXiv:1805.02628. Available from DOI: 10.48550/arXiv.1805.02628.

KASHYAP, Swati; SHARMA, Akshay; GAUTAM, Savit; SHARMA, Rishabh; CHAUHAN, Sneha; SIMRAN, 2024. Adversarial Attacks and Defenses in Deep Learning. In: *2024 International Conference on Emerging Innovations and Advanced Computing (INNOCOMP)* [online], pp. 318–323 [visited on 2024-12-19]. Available from DOI: 10.1109/INNOCOMP63224.2024.00059.

KHAMAISEH, Samer Y.; BAGAGEM, Derek; AL-ALAJ, Abdullah; MANCINO, Mathew; ALOMARI, Hakam W., 2022. Adversarial Deep Learning: A Survey on Adversarial Attacks and Defense Mechanisms on Image Classification. *IEEE Access* [online]. Vol. 10, pp. 102266–102291 [visited on 2024-12-19]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2022.3208131. Conference Name: IEEE Access.

KOH, Pang Wei; LIANG, Percy, 2017. Understanding Black-box Predictions via Influence Functions. In: [online] [visited on 2025-06-18]. Available from: https://www.semanticscholar.org/paper/Understanding-Black-box-Predictions-via-Influence-Koh-Liang/08ad8fad21f6ec4cda4d56be1ca5e146b7c913a1?utm_source=consensus.

LI, Deqiang; LI, Qianmu; YE, Yanfang; XU, Shouhuai, 2021. A Framework for Enhancing Deep Neural Networks Against Adversarial Malware. *IEEE Transactions on Network Science and Engineering* [online]. Vol. 8, no. 1, pp. 736–750 [visited on 2025-06-16]. ISSN 2327-4697. Available from DOI: 10.1109/TNSE.2021.3051354.

LI, M.; JIANG, P.; WANG, Q.; SHEN, C.; LI, Q., 2021. Adversarial Attacks and Defenses for Deep Learning Models. *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*. Vol. 58, no. 5, pp. 909–926. Available from DOI: 10.7544/issn1000-1239.2021.20200920.

LI, Yanjie; XIE, Bin; GUO, Songtao; YANG, Yuanyuan; XIAO, Bin, 2024. A Survey of Robustness and Safety of 2D and 3D Deep Learning Models against Adversarial Attacks. *ACM Comput. Surv.* [online]. Vol. 56, no. 6, 138:1–138:37 [visited on 2024-12-19]. ISSN 0360-0300. Available from DOI: 10.1145/3636551.

LI, Yudong; ZHANG, Shigeng; WANG, Weiping; SONG, Hong, 2023. Backdoor Attacks to Deep Learning Models and Countermeasures: A Survey. *IEEE Open Journal of the Computer Society* [online]. Vol. 4, pp. 134–146 [visited on 2025-04-16]. ISSN 2644-1268. Available from DOI: 10.1109/OJCS.2023.3267221.

LIU, X.; XIE, L.; WANG, Y.; LI, X., 2020. Adversarial attacks and defenses in deep learning. *Chinese Journal of Network and Information Security*. Vol. 6, no. 5, pp. 36–53. Available from DOI: 10.11959/j.issn.2096-109x.2020071.

LIU, Ximeng; XIE, Lehui; WANG, Yaopeng; ZOU, Jian; XIONG, Jinbo; YING, Zuobin; VASILAKOS, Athanasios V., 2021. Privacy and Security Issues in Deep Learning: A Survey. *IEEE Access* [online]. Vol. 9, pp. 4566–4593 [visited on 2024-11-02]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2020.3045078.

LIU, Yingqi; LEE, Wen-Chuan; TAO, Guanhong; MA, Shiqing; AAFER, Yousra; ZHANG, Xiangyu, 2019. ABS: Scanning Neural Networks for Back-doors by Artificial Brain Stimulation. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* [online]. New York, NY, USA: Association for Computing Machinery, pp. 1265–1282 [visited on 2025-06-18]. CCS '19. ISBN 978-1-4503-6747-9. Available from DOI: 10.1145/3319535.3363216.

MA, Xingjun; NIU, Yuhao; GU, Lin; WANG, Yisen; ZHAO, Yitian; BAILEY, James; LU, Feng, 2021. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition* [online]. Vol. 110, p. 107332 [visited on 2025-06-16]. ISSN 00313203. Available from DOI: 10.1016/j.patcog.2020.107332.

MACHOOKA, Daniel; YUAN, Xiaohong; ESTERLINE, Albert, 2023. A Survey of Attacks and Defenses for Deep Neural Networks. In: *2023 IEEE International Conference on Cyber Security and Resilience (CSR)* [online], pp. 254–261 [visited on 2024-11-02]. Available from DOI: 10.1109/CSR57506.2023.10224947.

OREKONDY, Tribhuvanesh; SCHIELE, Bernt; FRITZ, Mario, 2018. *Knockoff Nets: Stealing Functionality of Black-Box Models* [online]. arXiv [visited on 2025-06-18]. No. arXiv:1812.02766. Available from DOI: 10.48550/arXiv.1812.02766.

PENG, Haipeng; BAO, Shuang; LI, Lixiang, 2024. A Survey of Security Protection Methods for Deep Learning Model. *IEEE Transactions on Artificial Intelligence* [online]. Vol. 5, no. 4, pp. 1533–1553 [visited on 2024-11-02]. ISSN 2691-4581. Available from DOI: 10.1109/TAI.2023.3314398.

PRIYA, K V; DINESH, Peter J, 2023. A Detailed Study on Adversarial Attacks and Defense Mechanisms on Various Deep Learning Models. In: *2023 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)* [online], pp. 1–6 [visited on 2024-12-19]. Available from DOI: 10.1109/ACCTHPA57160.2023.10083378.

PUTTAGUNTA, Murali Krishna; RAVI, S.; NELSON KENNEDY BABU, C., 2023. Adversarial examples: attacks and defences on medical deep learning systems. *Multimedia Tools and Applications* [online]. Vol. 82, no. 22, pp. 33773–33809 [visited on 2025-06-16]. ISSN 1573-7721. Available from DOI: 10.1007/s11042-023-14702-9.

REN, K.; ZHENG, T.; QIN, Z.; LIU, X., 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering*. Vol. 6, no. 3, pp. 346–360. Available from DOI: 10.1016/j.eng.2019.12.012.

SHUVO, Md Shamimur Rahman; ALHADIDI, Dima, 2020. Membership Inference Attacks: Analysis and Mitigation. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* [online], pp. 1410–1419 [visited on 2025-06-18]. Available from DOI: 10.1109/TrustCom50675.2020.00190. ISSN: 2324-9013.

SUN, Guangling; SU, Yuying; QIN, Chuan; XU, Wenbo; LU, Xiaofeng; CEGLOWSKI, Andrzej, 2020. Complete Defense Framework to Protect Deep Neural Networks against Adversarial Examples. *Mathematical Problems in Engineering* [online]. Vol. 2020, no. 1, p. 8319249 [visited on 2025-06-16]. ISSN 1563-5147. Available from DOI: 10.1155/2020/8319249. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2020/8319249.

THANGARAJU, Arjun; MERKEL, Cory, 2022. Exploring Adversarial Attacks and Defenses in Deep Learning. In: *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* [online], pp. 1–6 [visited on 2024-12-19]. Available from DOI: 10.1109/CONECCT55679.2022.9865841. ISSN: 2766-2101.

WANG, Bolun; YAO, Yuanshun; SHAN, Shawn; LI, Huiying; VISWANATH, Bimal; ZHENG, Haitao; ZHAO, Ben Y., 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In: *2019 IEEE Symposium on Security and Privacy (SP)* [online], pp. 707–723 [visited on 2025-06-18]. Available from DOI: 10.1109/SP.2019.00031. ISSN: 2375-1207.

WANG, Chenxu; ZHANG, Ming; ZHAO, Jinjing; KUANG, Xiaohui, 2022. Black-Box Adversarial Attacks on Deep Neural Networks: A Survey. In: *2022 4th International Conference on Data Intelligence and Security (ICDIS)* [online], pp. 88–93 [visited on 2024-12-19]. Available from DOI: 10.1109/ICDIS55630.2022.00021.

XU, W.; EVANS, D.; QI, Y., 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In: available from DOI: 10.14722/ndss.2018.23198.

XUE, Mingfu; YUAN, Chengxiang; WU, Heyi; ZHANG, Yushu; LIU, Weiqiang, 2020. Machine Learning Security: Threats, Countermeasures, and Evaluations. *IEEE Access* [online]. Vol. 8, pp. 74720–74742 [visited on 2024-11-04]. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2020.2987435.

YE, Jiayuan; MADDI, Aadyaa; MURAKONDA, Sasi Kumar; BINDSCHAEDLER, Vincent; SHOKRI, Reza, 2022. Enhanced Membership Inference Attacks against Machine Learning Models. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* [online], pp. 3093–3106 [visited on 2025-06-18]. Available from DOI: 10.1145/3548606.3560675. Conference Name: CCS '22: 2022 ACM SIGSAC Conference on Computer and Communications Security ISBN: 9781450394505 Place: Los Angeles CA USA Publisher: ACM.

YE, Zipeng; LUO, Wenjian; NASEEM, Muhammad Luqman; YANG, Xiangkai; SHI, Yuhui; JIA, Yan, 2024. C2FMI: Corse-to-Fine Black-Box Model Inversion Attack. *IEEE Transactions on Dependable and Secure Computing* [online]. Vol. 21, no. 3, pp. 1437–1450 [visited on 2025-06-18]. ISSN 1941-0018. Available from DOI: 10.1109/TDSC.2023.3285071.

YUAN, Xiaoyong; HE, Pan; ZHU, Qile; LI, Xiaolin, 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems* [online]. Vol. 30, no. 9, pp. 2805–2824 [visited on 2024-12-19]. ISSN 2162-2388. Available from DOI: 10.1109/TNNLS.2018.2886017. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

ZHOU, Shuai; LIU, Chi; YE, Dayong; ZHU, Tianqing; ZHOU, Wanlei; YU, Philip S., 2022. Adversarial Attacks and Defenses in Deep Learning: From a Perspective of Cybersecurity. *ACM Comput. Surv.* [online]. Vol. 55, no. 8, 163:1–163:39 [visited on 2024-12-19]. ISSN 0360-0300. Available from DOI: 10.1145/3547330.

ZHOU, Shuai; ZHU, Tianqing; YE, Dayong; YU, Xin; ZHOU, Wanlei, 2024. Boosting Model Inversion Attacks With Adversarial Examples. *IEEE Transactions on Dependable and Secure Computing* [online]. Vol. 21, no. 3, pp. 1451–1468 [visited on 2025-06-18]. ISSN 1545-5971, ISSN 1941-0018, ISSN 2160-9209. Available from DOI: 10.1109/TDSC.2023.3285015.

ZHOU, Yiyun; HAN, Meng; LIU, Liyuan; HE, Jing; GAO, Xi, 2019. The Adversarial Attacks Threats on Computer Vision: A Survey. In: *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)* [online], pp. 25–30 [visited on 2024-11-03]. Available from DOI: 10.1109/MASSW.2019.00012.

ZHU, Tianqing; YE, Dayong; ZHOU, Shuai; LIU, Bo; ZHOU, Wanlei, 2023. Label-Only Model Inversion Attacks: Attack With the Least Information. *IEEE Transactions on Information Forensics and Security* [online]. Vol. 18, pp. 991–1005 [visited on 2025-06-18]. ISSN 1556-6021. Available from DOI: 10.1109/TIFS.2022.3233190.

Appendix A

Systematic Literature Review Protocol

This appendix provides a detailed account of the systematic literature review methodology applied to support the analysis presented in Chapter 2. The review process followed a PRISMA-inspired strategy, adapted to the needs of a software engineering and machine learning context. The objective was to ensure a rigorous, transparent, and reproducible selection of scientific literature relevant to adversarial threats and mitigation strategies in DL.

The core of the process involved structured queries applied to major digital libraries: IEEE Xplore, Scopus, and the ACM Digital Library. These were designed to retrieve peer-reviewed and high-quality studies aligned with the dissertation's research scope. Additionally, a complementary set of relevant studies was identified through exploratory searches using the AI-assisted Consensus platform. This step enhanced literature coverage, particularly for papers referenced in surveys or not captured by the initial keyword-based queries.

Furthermore, a small number of foundational or highly influential works were manually included due to their relevance and frequent citation across the literature. Some of these were also suggested by the dissertation supervisors based on domain expertise and alignment with the research goals.

This appendix presents the full protocol of the review, including the search strategy, execution of queries, inclusion and exclusion criteria, the screening and selection process, and a summary of the results supported by a PRISMA-style flow diagram.

A.1 Search Strategy

The search strategy was designed to ensure broad yet focused coverage of scientific publications relevant to adversarial threats and defence mechanisms in deep learning. To achieve this, structured keyword-based queries were formulated and applied across three well-established digital libraries widely used in engineering and computer science: IEEE Xplore, Scopus, and the ACM Digital Library.

The search was conducted exclusively within article titles to ensure the retrieval of studies with direct thematic alignment, and to avoid the excessive noise often introduced by full-text or

abstract-level searches. To ensure the relevance and currency of the review, only publications from 2018 onwards were considered. This cut-off reflects the rapid evolution of adversarial machine learning, and ensures that the included works capture recent advancements and trends.

Each database was queried independently using a consistent set of queries (presented in Section A.2), adapted where necessary to match the syntax and features of each digital platform. In all cases, filters were applied to restrict results to peer-reviewed journal articles, conference papers, and book chapters published in English.

In addition to these structured searches, complementary exploratory searches were performed using the Consensus platform, an AI-powered academic search tool. This was particularly useful for identifying recent or highly cited studies referenced in survey papers or related works that were not retrieved via the initial queries. The integration of Consensus into the search process improved recall and allowed for the discovery of emerging research directions beyond the scope of predefined keyword combinations.

A small number of additional studies were also included manually based on their foundational status in the field or on the recommendation of the dissertation supervisors. These additions were carefully reviewed to ensure consistency with the overall inclusion criteria.

A.2 Search Queries and Execution

To ensure comprehensive coverage of relevant literature, four distinct search queries were designed. Each query targeted a specific subset of the adversarial machine learning domain, covering attacks, defences, detection mechanisms, and robustness techniques in DL systems. The queries were designed to maximize relevance while minimizing redundancy, and were executed across all three digital libraries listed in Section A.1.

Table A.1 summarizes the queries used in this review, along with a brief justification for each.

Each query was executed individually on IEEE Xplore, Scopus, and the ACM Digital Library, respecting the same filters across platforms: only peer-reviewed studies published in English from 2018 onwards were considered. Title-based searches were used in all platforms to maintain consistency and ensure direct thematic relevance.

A summary of the number of records retrieved per query and per database, prior to any screening or duplicate removal, is presented in Table A.2. These values reflect the initial stage of the review pipeline.

These figures do not account for duplicates across databases or additional studies identified through manual inspection and the Consensus platform, which are addressed in Section A.4.

A.3 Inclusion and Exclusion Criteria

To ensure the quality, relevance, and scientific rigour of the selected studies, a structured set of inclusion and exclusion criteria was defined prior to the screening process. These criteria were

Table A.1: Search Queries and Justifications

Query ID	Search Query and Justification
Q1	**Query:** ("adversarial attacks" OR "adversarial examples") AND ("deep learning" OR "neural networks") *Justification:* Captures general studies focusing on adversarial threats applied to neural networks and DL models.
Q2	**Query:** ("defence" OR "defence" OR "security" OR "mitigation strategies") AND ("attack") AND ("deep learning" OR "neural networks") *Justification:* Focuses on works addressing defence mechanisms or mitigation strategies against adversarial threats in DL.
Q3	**Query:** ("detection") AND ("adversarial attacks" OR "adversarial examples") AND ("deep learning" OR "neural networks") *Justification:* Targets approaches focused on the detection of adversarial inputs.
Q4	**Query:** ("adversarial training" OR "ensemble methods") AND ("deep learning" OR "neural networks") *Justification:* Highlights literature related to training techniques aimed at improving robustness against adversarial examples.

applied uniformly to all records retrieved via structured queries, as well as those obtained through manual selection and the Consensus platform.

Only records that met *all* inclusion criteria were retained for full-text analysis and integration into the review. Records that failed to meet any of the criteria listed in Table A.3 were excluded from further consideration.

The application of these criteria ensured that only scientifically robust, peer-reviewed, and thematically aligned studies were included in the final literature set. This process enhanced the consistency and reliability of the review outcomes.

A.4 Screening and Selection Process

Following the initial search and retrieval phase, a structured screening and selection process was conducted to refine the pool of candidate publications. This process involved the elimination of duplicates, the application of inclusion and exclusion criteria, and the verification of full-text availability.

Initial Pool of Records

A total of 664 records were retrieved through structured queries applied to IEEE Xplore, Scopus, and the ACM Digital Library, as summarized in Table A.2. In addition to these, 37 supplementary records were included from alternative sources:

- 24 articles identified through exploratory searches using the AI-assisted Consensus platform;

Table A.2: Search Results by Query and Digital Library

Database	Q1	Q2	Q3	Q4	Total
IEEE Xplore	177	24	0	34	235
Scopus	209	85	18	80	392
ACM Digital Library	29	0	3	5	37
Total	**415**	**109**	**21**	**119**	**664**

- 13 articles manually selected based on their foundational relevance or supervisor recommendation.

This resulted in an initial pool of **701 records** prior to duplicate removal.

Duplicate Removal and Initial Screening

After merging the results from all sources, a duplicate removal step was applied, reducing the set to **555 unique records**. This step ensured that each publication was assessed only once.

The remaining records were then screened by applying the inclusion and exclusion criteria defined in Section A.3. Each paper was examined individually to determine if it met *all* of the required criteria. As a result:

- **56 studies** satisfied all inclusion criteria and were retained for full-text review;
- **499 records** were excluded for failing to meet at least one of the inclusion criteria.

PRISMA Flow Diagram

Figure A.1 summarizes the full screening and selection pipeline using a PRISMA-style diagram.

This multi-stage process ensured that only high-quality, relevant, and accessible studies were included in the review. The resulting **56 publications** form the scientific foundation for the analysis and conclusions presented in the main body of the dissertation.

Table A.3: Inclusion Criteria for Literature Selection

ID	Criterion Description
C1	Result of executing a valid search query or identified manually (via Consensus platform or expert recommendation).
C2	Published in or after 2018.
C3	Full-text access available.
C4	Written in English.
C5	Explicitly addresses topics related to adversarial attacks, adversarial examples, data poisoning, model extraction, defences, security measures, or detection mechanisms in DL or neural networks.
C6	Provides sufficient depth, originality, and relevance to the research objectives.
C7	Not duplicated across data sources.
C8	Not retracted at the time of review.
C9	Not a work-in-progress, poster, or extended abstract.
C10	Scholarly publication: journal article, conference paper, or book chapter.
C11	Not a summary of a tutorial, workshop, lecture, or seminar.

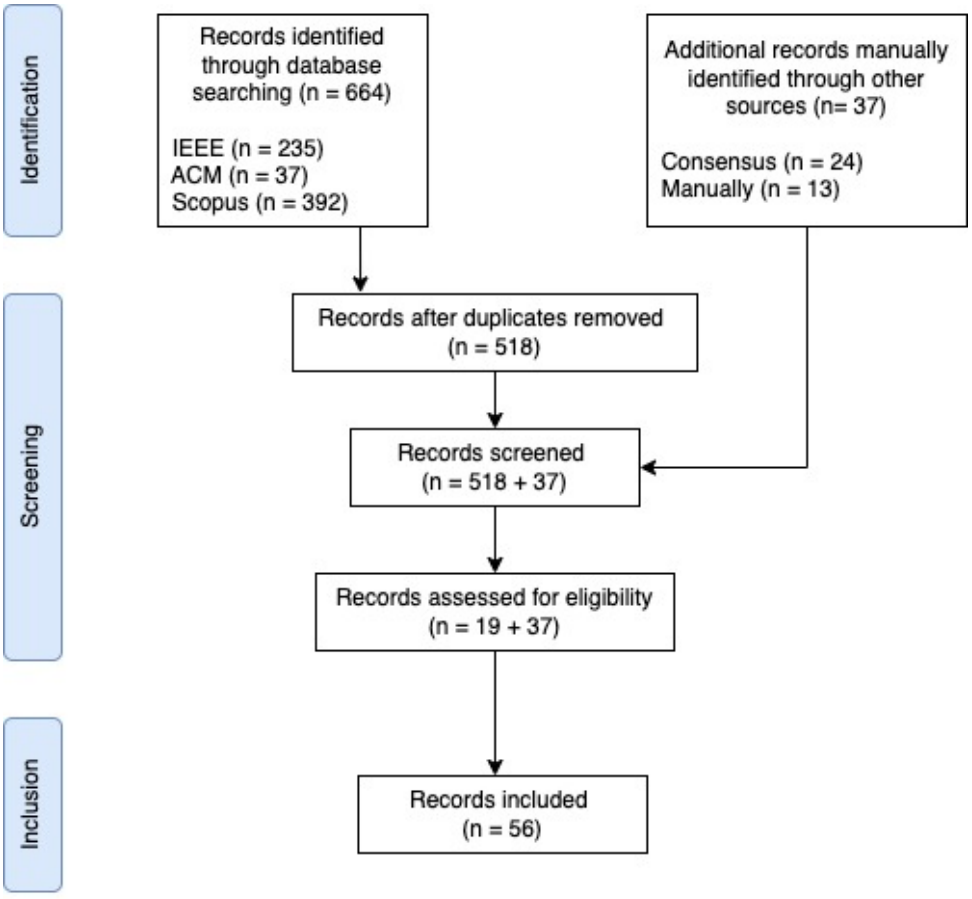


Figure A.1: PRISMA flow diagram illustrating the literature screening and selection process

Appendix B

Relevant Outputs from Framework Execution

This section collects visual excerpts and generated artifacts from the execution of the `Safe-DL` framework, highlighting important steps or results obtained through the CLI or internal processes.

B.1 Module 1: Threat Modelling

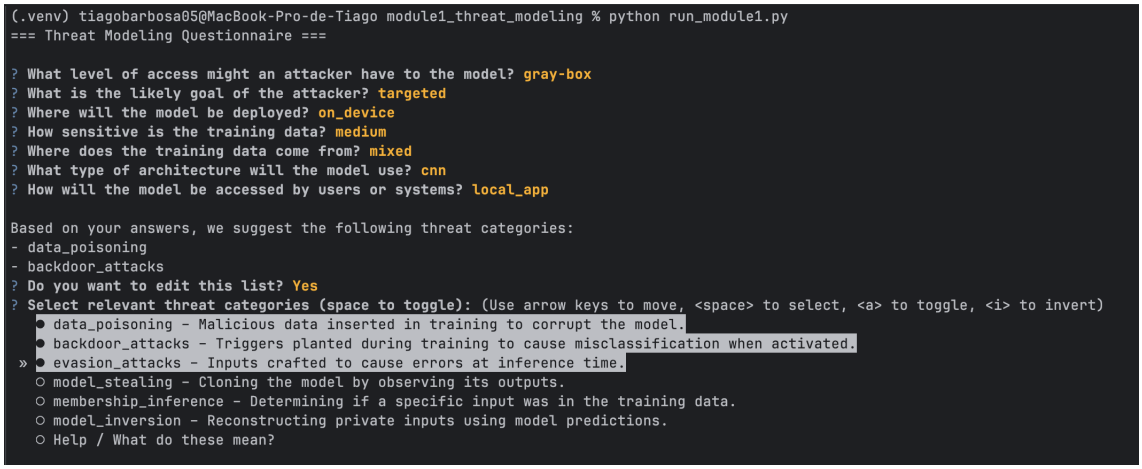


Figure B.1: Questionnaire responses and automatically suggested threat categories.

```
Based on your answers, we suggest the following threat categories:
- data_poisoning
- backdoor_attacks
? Do you want to edit this list? Yes
? Select relevant threat categories (space to toggle): done (3 selections)

=== Generated Threat Profile ===
threat_model:
  model_access: gray-box
  attack_goal: targeted
  deployment_scenario: on_device
  data_sensitivity: medium
  training_data_source: mixed
  model_type: cnn
  interface_exposed: local_app
  threat_categories:
  - data_poisoning
  - backdoor_attacks
  - evasion_attacks

? Do you want to save this profile to a YAML file? Yes
? Enter filename (e.g., 'test' to save as test.yaml): visitech
```

Figure B.2: Final YAML threat profile generated based on the selected options.

B.2 Module 2: Attack Simulation

```
(.venv) tiagobarbosa05@MacBook-Pro-de-Tiago module2_attack_simulation % python setup_module2.py

=== Safe-DL Framework – Module 2 Setup Wizard ===

? Select a dataset: cifar10 (built-in)
Files already downloaded and verified
Files already downloaded and verified
? Select a model: resnet18
? Do you want to customize the input shape? No
? Select a threat profile to use: ../profiles/visitech.yaml
[*] Data poisoning enabled in threat model.
? Select the data poisoning attacks to simulate: (Use arrow keys to move, <space> to select, <a> to toggle, <i> to invert)
 » ● Label Flipping
   ○ Clean Label
```

Figure B.3: Execution step 1 from Module 2 CLI pipeline.

```
=== Configuring Label Flipping ===

Suggested strategy: many_to_one
Suggested flip_rate: 0.08
Suggested target class: 3 - cat
? Do you want to accept these suggestions? No
? Choose flipping strategy: Random to fixed (many→one)
? Flip rate (e.g., 0.08): 0.08
? Pick target class randomly? No
? Select target class (flip TO): 9 - truck
[*] Data poisoning configuration: {'label_flipping': {'strategy': 'many_to_one', 'flip_rate': 0.08, 'source_class': None, 'target_class': 9}}
[*] Backdoor attacks enabled in threat model.
? Select the backdoor attacks to simulate: [Static Patch Trigger]

=== Configuring Static Patch Backdoor Attack ===

Suggested configuration:
  - Target class: 2 - bird
  - Patch type: white_square
  - Patch size (relative): 0.15
  - Poison fraction: 0.05
  - Patch position: bottom_right
  - Label mode: corrupted
  - Blend alpha: 1.0
? Do you want to accept these suggestions? No
```

Figure B.4: Execution step 2 from Module 2 CLI pipeline.

```
Suggested configuration:
  - Target class: 2 - bird
  - Patch type: white_square
  - Patch size (relative): 0.15
  - Poison fraction: 0.05
  - Patch position: bottom_right
  - Label mode: corrupted
  - Blend alpha: 1.0
? Do you want to accept these suggestions? No
? Pick target class randomly? No
? Select target class (to misclassify as): 7 - horse
? Select the type of patch to apply: white_square
? Select where to place the patch: bottom_right
? Patch size (as ratio of image width, e.g., 0.2): 0.15
? Poisoning fraction (e.g., 0.1): 0.15
? Select label mode: corrupted
? Blending alpha (0.0 to 1.0): 1.0
[*] Backdoor configuration: {'static_patch': {'target_class': 7, 'patch_type': 'white_square', 'patch_position': 'bottom_right', 'patch_size_ratio': 0.15, 'poison_frac
tion': 0.15, 'label_mode': 'corrupted', 'blend_alpha': 1.0}}
[*] Evasion attacks enabled in threat model.
? Select the evasion attacks to simulate: (Use arrow keys to move, <space> to select, <a> to toggle, <i> to invert)
   ○ Fast Gradient Sign Method (FGSM)
   ● Projected Gradient Descent (PGD)
   ○ Carlini & Wagner (C&W)
   ○ DeepFool
   ○ Natural Evolution Strategies (NES)
 » ● Simultaneous Perturbation Stochastic Approximation (SPSA)
   ○ Transfer-based Attack
```

Figure B.5: Execution step 3 from Module 2 CLI pipeline.

```
=== Configuring PGD Attack ===

Suggested configuration:
- Epsilon      : 0.03
- Alpha        : 0.01
- Num Iterations: 50
- Random Start : True
  • Max samples to attack         : 1000
? Do you want to accept these suggestions? Yes

=== Configuring SPSA Attack ===

Suggested configuration based on threat model:
  • Epsilon (max perturbation)    : 0.05
  • Delta (perturbation size)     : 0.01
  • Learning rate                 : 0.01
  • Number of optimization steps  : 100
  • Batch size for gradient estimate: 32
  • Max samples to attack         : 500
? Do you want to accept these suggested values? No
? Epsilon (e.g., 0.05): 0.03
? Delta (perturbation size for SPSA, e.g., 0.01): 0.01
? Learning rate (e.g., 0.01): 0.01
? Number of steps (e.g., 100): 150
? Batch size (e.g., 32): 32
? Max samples to attack (0 = all): 500

[✓] Profile updated and saved at: ../profiles/visitech.yaml
```

Figure B.6: Execution step 4 from Module 2 CLI pipeline.

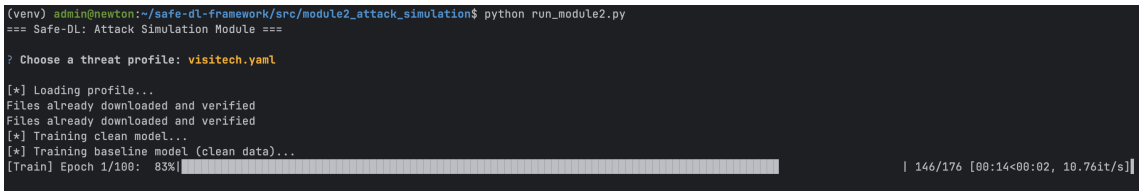


Figure B.7: Execution step 5 from Module 2 CLI pipeline.

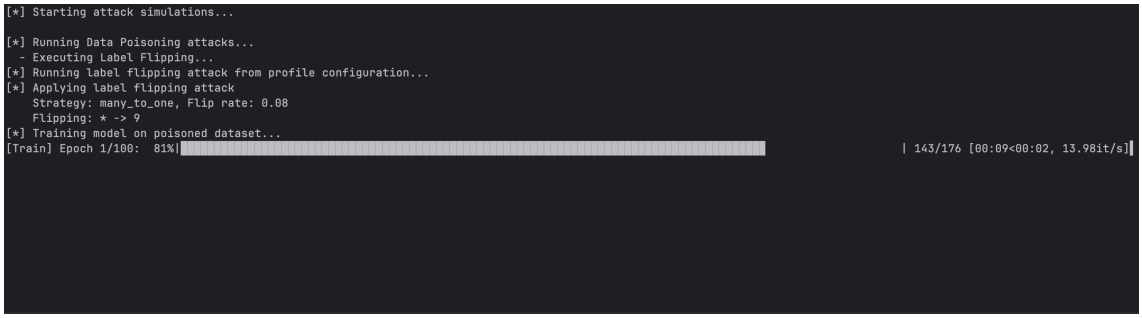


Figure B.8: Execution step 6 from Module 2 CLI pipeline.

Figure B.9: Execution step 7 from Module 2 CLI pipeline.

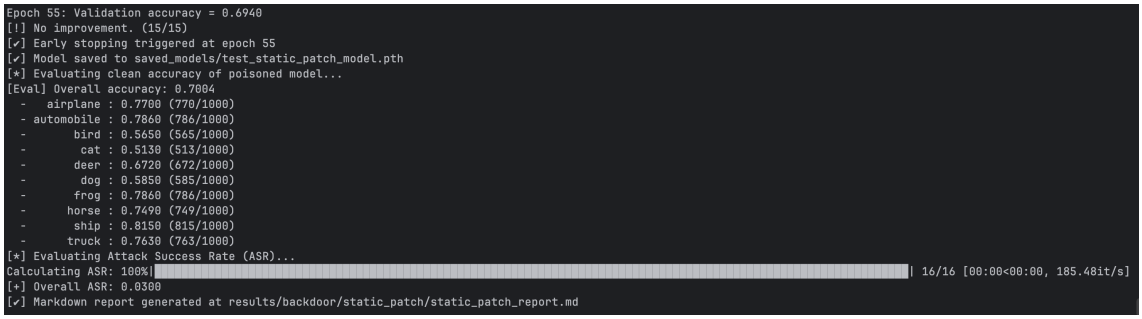


Figure B.10: Execution step 8 from Module 2 CLI pipeline.

Figure B.11: Execution step 9 from Module 2 CLI pipeline.

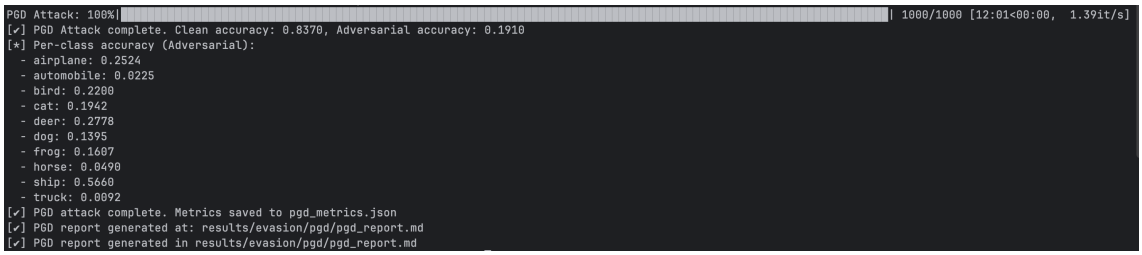


Figure B.12: Execution step 10 from Module 2 CLI pipeline.

```
# Data Poisoning - Label Flipping Attack Report

## Overview

- **Attack Type:** label_flipping
- **Strategy:** many_to_one
- **Flip Rate:** 0.08
- **Target Class:** 9
- **Source Classes:** All except target
- **Number of Flipped Samples:** 3240

## Performance Metrics

- **Accuracy After Attack:** 0.6578

### Per-Class Accuracy

| Class | Accuracy |
|--------|----------|
| airplane | 0.7110 |
| automobile | 0.7080 |
| bird | 0.5050 |
| cat | 0.4180 |
| deer | 0.6010 |
| dog | 0.5890 |
| frog | 0.7570 |
| horse | 0.7120 |
| ship | 0.7780 |
| truck | 0.7990 |

## Flip Summary

| Original -> New | Count |
|------------------|--------|
| ship->truck | 361 |
| automobile->truck | 363 |
| dog->truck | 384 |
| bird->truck | 362 |
```

```
| frog->truck | 332 |
| deer->truck | 353 |
| airplane->truck | 349 |
| cat->truck | 347 |
| horse->truck | 389 |

## Example Flips

| Index | Original Label | New Label |
|--------|----------------|-----------|
| 39442 | ship | truck |
| 26239 | ship | truck |
| 13006 | automobile | truck |
| 17898 | dog | truck |
| 18331 | automobile | truck |

## Visual Flip Examples (first 5)

<div style="display: flex; gap: 10px;">
<div style="text-align: center;"><small><strong>ship -> truck</strong></small><br><
    img src="examples/flip_39442_8_to_9.png" alt="flip" style="width: 120px;"></div
    >
<div style="text-align: center;"><small><strong>ship -> truck</strong></small><br><
    img src="examples/flip_26239_8_to_9.png" alt="flip" style="width: 120px;"></div
    >
<div style="text-align: center;"><small><strong>automobile -> truck</strong></small
    ><br><img src="examples/flip_13006_1_to_9.png" alt="flip" style="width: 120px;"
    ></div>
<div style="text-align: center;"><small><strong>dog -> truck</strong></small><br><
    img src="examples/flip_17898_5_to_9.png" alt="flip" style="width: 120px;"></div
    >
<div style="text-align: center;"><small><strong>automobile -> truck</strong></small
    ><br><img src="examples/flip_18331_1_to_9.png" alt="flip" style="width: 120px;"
    ></div>
</div>
```

Listing B.1: Label Flipping Markdown Report.

Figure B.13: Example image affected by Label Flipping: `automobile → truck`.

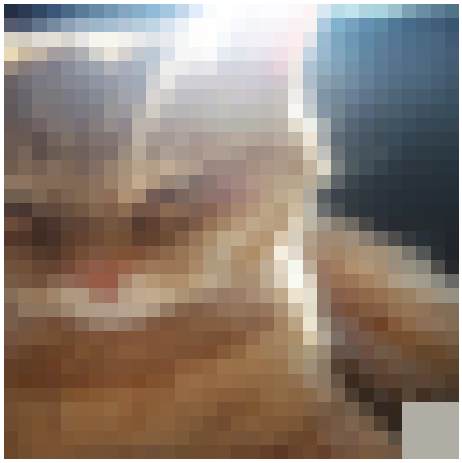


Figure B.14: Example image with visible backdoor (Static Patch Attack).

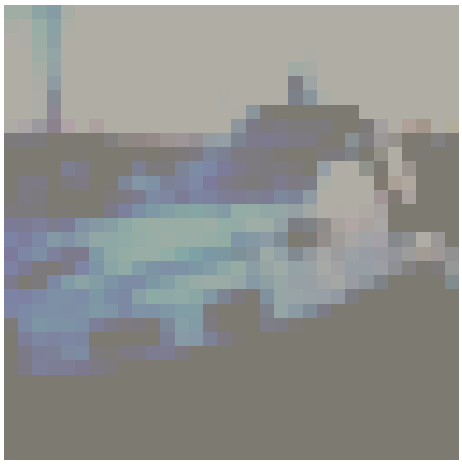


Figure B.15: Example of successful misclassification under PGD evasion.

B.3 Module 3: Risk Analysis

```
(venv) admin@newton:~/safe-dl-framework/src/module3_risk_analysis$ python run_module3.py
? Select a threat profile to use: ../profiles/visitech.yaml
[*] Analyzing attack: Label Flipping
[*] Analyzing attack: Static Patch
[*] Analyzing attack: PGD
[*] Analyzing attack: SPSA

[✓] Risk analysis saved to results/risk_analysis.json
[✓] Risk report saved to: results/risk_report.md
[✓] Recommendations saved to: ../profiles/visitech.yaml under `risk_analysis.recommendations`
[✓] Summary saved to: ../profiles/visitech.yaml under `risk_analysis.summary`
```

Figure B.16: CLI trace of Module 3 execution, showing attack-specific risk computation and storage of analysis artifacts.

B.4 Module 4: Defense Application

```
=== Safe-DL — Module 4 Setup Wizard ===

? Select a threat profile: ../profiles/visitech.yaml
[*] Configuring defenses for Data Poisoning...

=== LABEL_FLIPPING ===
Suggested defenses: ['data_cleaning', 'per_class_monitoring']
? Accept all suggested defenses? No
? Select defenses to apply: [data_cleaning — Remove corrupted or mislabeled samples using heuristics, clustering, or manual review.]
[?] Configuring parameters for defense: data_cleaning
? Cleaning strategy: loss_filtering
? Anomaly threshold (e.g., 0.9): 0.9

=== CLEAN_LABEL ===
Suggested defenses: []
? Accept all suggested defenses? Yes
[*] Configuring defenses for Evasion...

=== FGSM ===
Suggested defenses: []
? Accept all suggested defenses? Yes

=== PGD ===
Suggested defenses: ['adversarial_training', 'randomized_smoothing']
? Accept all suggested defenses? (Y/n)
```

Figure B.17: Module 4 setup CLI interaction — Part 1.

```
=== PGD ===
Suggested defenses: ['adversarial_training', 'randomized_smoothing']
? Accept all suggested defenses? Yes
[?] Configuring parameters for defense: adversarial_training
? Base attack for adversarial training: fgsm
? Training epsilon: 0.03
[?] Configuring parameters for defense: randomized_smoothing
? Noise level σ (e.g. 0.25): 0.25

=== NES ===
Suggested defenses: []
? Accept all suggested defenses? Yes

=== SPSA ===
Suggested defenses: ['gradient_masking', 'jpeg_preprocessing']
? Accept all suggested defenses? Yes
[?] Configuring parameters for defense: gradient_masking
? Gradient masking strength (0.0-1.0): 0.5
[?] Configuring parameters for defense: jpeg_preprocessing
? JPEG compression quality (1-100): 75

=== CW ===
Suggested defenses: []
? Accept all suggested defenses? Yes
```

Figure B.18: Module 4 setup CLI interaction — Part 2.

```
=== DEEPFOOL ===
Suggested defenses: []
? Accept all suggested defenses? Yes

=== TRANSFER ===
Suggested defenses: []
? Accept all suggested defenses? Yes
[*] Configuring defenses for Backdoor...

=== STATIC_PATCH ===
Suggested defenses: ['activation_clustering', 'spectral_signatures']
? Accept all suggested defenses? Yes
[?] Configuring parameters for defense: activation_clustering
? Number of clusters: 2
[?] Configuring parameters for defense: spectral_signatures
? SVD threshold (0-1): 0.9

=== LEARNED_TRIGGER ===
Suggested defenses: []
? Accept all suggested defenses? Yes

[✓] Profile updated with defenses and saved to: ../profiles/visitech.yaml
```

Figure B.19: Module 4 setup CLI interaction — Part 3.

Figure B.20: CLI output of label flipping defense using `data_cleaning`.

Figure B.21: CLI output of backdoor mitigation with `activation_clustering`.

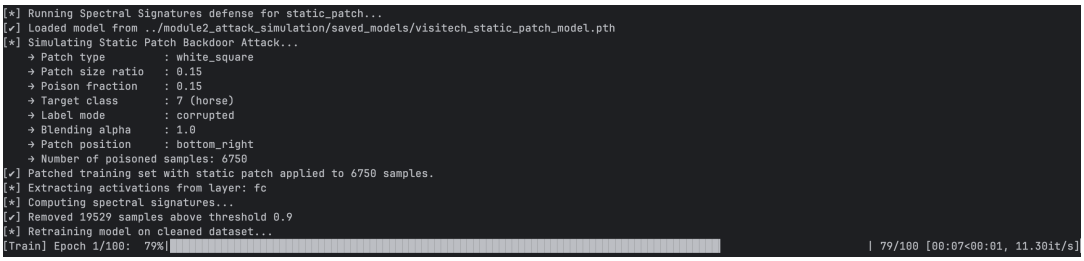


Figure B.22: CLI output of backdoor mitigation with `spectral_signatures`.

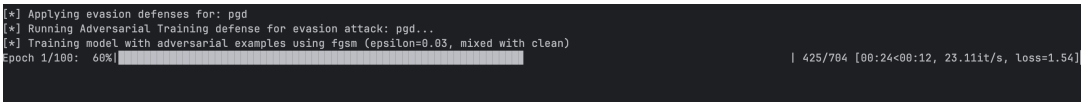


Figure B.23: CLI output of adversarial training for PGD attack.

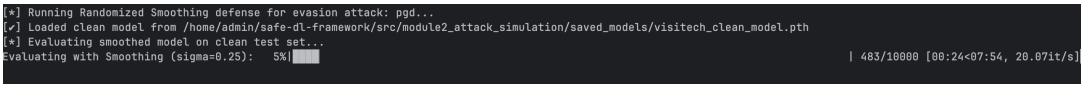


Figure B.24: CLI output of evaluation with randomized smoothing.

```
[*] Applying evasion defenses for: spsa
[*] Running Gradient Masking defense for evasion attack: spsa...
[✓] Loaded clean model from /home/admin/safe-dl-framework/src/module2_attack_simulation/saved_models/visitech_clean_model.pth
[*] Evaluating masked model on clean test set...
Evaluating with gradient masking: 100%|████████████████████████████████████████████████████████████████████████████████| 157/157 [00:03<00:00, 49.91it/s]
[*] Generating adversarial examples with SPSA...
[*] Applying SPSA attack to dataset...
SPSA Attack:   2%|▌                                                                                                | 12/500 [00:13<10:03,  1.24s/it]
```

Figure B.25: CLI output of gradient masking and SPSA evaluation.

```
[*] Running JPEG Preprocessing defense for evasion attack: spsa...
[✓] Loaded clean model from /home/admin/safe-dl-framework/src/module2_attack_simulation/saved_models/visitech_clean_model.pth
[*] Evaluating JPEG-transformed model on clean test set...
Clean Eval: 100%|███████████████████████████████████████████████████████████████████████████████████████████████| 157/157 [00:07<00:00, 22.27it/s]
[*] Generating adversarial examples...
[*] Applying SPSA attack to dataset...
SPSA Attack:   3%|▊                                                                                                | 15/500 [00:14<08:09,  1.01s/it]
```

Figure B.26: CLI output of JPEG preprocessing and SPSA evaluation.

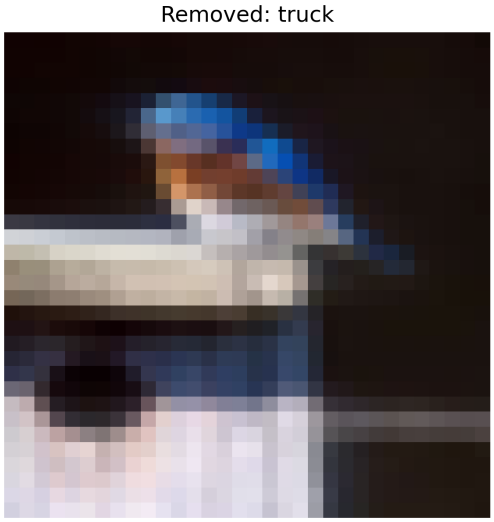


Figure B.27: Example of a removed image with activation clustering

Figure B.28: Example of a removed image with spectral signatures

```
1
2  # Defense Report     Data Cleaning
3
4  ## Overview
5
6  - **Defense:** data_cleaning
7  - **Attack Targeted:** label_flipping
8  - **Cleaning Method:** loss_filtering
9  - **Threshold:** 0.9
10
11 ## Performance Metrics
12
13 - **Accuracy After Defense:** 0.6837
14
15 ###  P e r Class  Accuracy
16
17 | Class | Accuracy |
18 |-------|----------|
19 | airplane | 0.7490 |
20 | automobile | 0.7640 |
21 | bird | 0.5540 |
22 | cat | 0.5070 |
23 | deer | 0.6400 |
24 | dog | 0.5520 |
25 | frog | 0.7660 |
26 | horse | 0.7290 |
27 | ship | 0.8110 |
28 | truck | 0.7650 |
29
30 ## Cleaning Summary
```

```
31
32 - **Total Samples Removed:** 6967
33
34 ## Example Removed Samples
35
36 The following examples illustrate removed samples identified as outliers or noisy
       instances.
37 Each image is named using the format:
38
39 ```
40 removed_<index>_<label>.png
41 ```
42 - `<index>`: Sample index in the dataset.
43 - `<label>`: Original class label.
44
45 <div style="display: flex; gap: 10px; flex-wrap: wrap;">
46 <div style="text-align:center;"><small>cleaned_examples/removed_32769_5.png</small
       ><br><img src="cleaned_examples/removed_32769_5.png" style="width: 120px;"></
       div>
47 <div style="text-align:center;"><small>cleaned_examples/removed_6_2.png</small><br
       ><img src="cleaned_examples/removed_6_2.png" style="width: 120px;"></div>
48 <div style="text-align:center;"><small>cleaned_examples/removed_32775_0.png</small
       ><br><img src="cleaned_examples/removed_32775_0.png" style="width: 120px;"></
       div>
49 <div style="text-align:center;"><small>cleaned_examples/removed_32779_7.png</small
       ><br><img src="cleaned_examples/removed_32779_7.png" style="width: 120px;"></
       div>
50 <div style="text-align:center;"><small>cleaned_examples/removed_13_9.png</small><br
       ><img src="cleaned_examples/removed_13_9.png" style="width: 120px;"></div>
51 </div>
```

Listing B.2: `data_cleaning` Markdown report.

B.5 Module 5: Defense Evaluation

```
(.venv) tiagobarbosa05@MacBook-Pro-de-Tiago module5_defense_evaluation % python run_module5.py
? Select a threat profile to use: ../profiles/visitech.yaml
[+] Baseline Accuracy: 0.8293

[+] Attack Category: backdoor
  - Attack: static_patch
    > Defenses applied: ['activation_clustering', 'spectral_signatures']
      - Checking: ../module4_defense_application/results/backdoor/static_patch/activation_clustering_results.json
      [✓] Scores for activation_clustering:
         Mitigation Score : 0.97
         CAD Score        : 0.887
         Cost Score       : 0.3
         Final Score      : 0.926
      - Checking: ../module4_defense_application/results/backdoor/static_patch/spectral_signatures_results.json
      [✓] Scores for spectral_signatures:
         Mitigation Score : 0.0
         CAD Score        : 0.743
         Cost Score       : 0.5
         Final Score      : 0.142

[+] Attack Category: data_poisoning
  - Attack: label_flipping
    > Defenses applied: ['data_cleaning']
      - Checking: ../module4_defense_application/results/data_poisoning/label_flipping/data_cleaning_results.json
      [✓] Scores for data_cleaning:
         Mitigation Score : 0.151
         CAD Score        : 1.039
         Cost Score       : 0.2
         Final Score      : 0.322
```

Figure B.29: Module 5 CLI output (part 1): Evaluation of backdoor and data poisoning defenses.

```
[+] Attack Category: evasion
  - Attack: pgd
    > Defenses applied: ['adversarial_training', 'randomized_smoothing']
      - Checking: ../module4_defense_application/results/evasion/pgd/adversarial_training_results.json
      [✓] Scores for adversarial_training:
         Mitigation Score : 0.282
         CAD Score        : 0.923
         Cost Score       : 0.8
         Final Score      : 0.38
      - Checking: ../module4_defense_application/results/evasion/pgd/randomized_smoothing_results.json
      [✓] Scores for randomized_smoothing:
         Mitigation Score : 0.01
         CAD Score        : 0.449
         Cost Score       : 0.5
         Final Score      : 0.093
  - Attack: spsa
    > Defenses applied: ['gradient_masking', 'jpeg_preprocessing']
      - Checking: ../module4_defense_application/results/evasion/spsa/gradient_masking_results.json
      [✓] Scores for gradient_masking:
         Mitigation Score : 0.006
         CAD Score        : 0.998
         Cost Score       : 0.4
         Final Score      : 0.196
      - Checking: ../module4_defense_application/results/evasion/spsa/jpeg_preprocessing_results.json
      [✓] Scores for jpeg_preprocessing:
         Mitigation Score : 0.003
         CAD Score        : 0.144
         Cost Score       : 0.1
         Final Score      : 0.031

[✓] All scores saved to results/defense_evaluation.json
[✓] Report generated at: results/defense_evaluation_report.md
```

Figure B.30: Module 5 CLI output (part 2): Evaluation of evasion defenses and final report generation.

B.6 Module 6: Final Reporting

```
(venv) admin@newton:~/safe-dl-framework/src/module6_reporting$ python run_module6.py
? Select a threat profile to use for the final report: visitech.yaml

[INFO] Generating final report for profile: visitech.yaml

[✓] Final report generated successfully at: ../reports/final_report.md
```

Figure B.31: CLI execution of Module 6 showing the final report generation.

```
1
2  # Safe-DL Framework - Final Security Report
3  **Profile Selected**: `visitech.yaml`
4  **Report Generated On**: 2025-06-23 16:42:23
5
6  ---
7
8  ## 1. Introduction and Overview
9  This comprehensive report aggregates the findings from the Safe-DL framework's
       security assessment, covering threat modeling, attack simulation, risk analysis
       , defense application, and defense evaluation. Its purpose is to provide a
       unified overview of the deep learning model's security posture against
       adversarial threats, document the mitigation strategies applied, and quantify
       their effectiveness. This dossier serves as a critical resource for decision-
       making regarding model deployment and continuous security improvement.
10
11 ## 2. System Under Evaluation Details
12 This section details the core components of the system analyzed in this report, as
       defined in the selected threat profile.
13
14 ### 2.1 Model Details
15 - **Name**: `resnet18`
16 - **Type**: `builtin`
17 - **Input Shape**: `[3, 32, 32]`
18 - **Number of Classes**: `10`
19
20
21 ### 2.2 Dataset Details
22 - **Name**: `cifar10`
23 - **Type**: `builtin`
24
25 ## 3. Threat Profile (Module 1)
26 This section outlines the specific characteristics of the system's environment and
       the anticipated adversary, as defined during the threat modeling phase (Module
       1). These parameters guide the subsequent attack simulations, risk analysis,
       and defense applications.
27
28 - **Model Access**: `white-box`
```

```
  *Describes the level of access the adversary has to the model internals (e.g., weights, architecture).*

- **Attack Goal**: `targeted`
  *Defines the adversary's objective (e.g., targeted misclassification, untargeted denial of service).*

- **Deployment Scenario**: `cloud`
  *Indicates where the model is deployed (e.g., cloud, edge device, mobile).*

- **Interface Exposed**: `api`
  *How the model interacts with external entities (e.g., API, direct access, web application).*

- **Model Type**: `cnn`
  *The architectural type of the deep learning model.*

- **Data Sensitivity**: `high`
  *The sensitivity level of the data used by the model, impacting privacy concerns.*

- **Training Data Source**: `internal_clean`
  *Origin and cleanliness of the data used for training the model.*

- **Threat Categories**:
    - `data_poisoning`
    - `backdoor_attacks`
    - `evasion_attacks`
    - `model_stealing`
    - `membership_inference`
    - `model_inversion`

  *A list of attack types considered relevant for this threat profile.*

*Note: While listed in the threat profile, 'model_stealing', 'membership_inference', and 'model_inversion' attack simulations and their corresponding defenses are currently considered future work and are not yet fully implemented in subsequent Modules.*

## 4. Attack Simulation (Module 2)
This section summarizes the outcomes of the adversarial attack simulations performed against the model based on the defined threat profile. These simulations quantify the model's vulnerability to various attack types before any defenses are applied.

### 4.1 Overview of Simulated Attacks

| Attack Category | Attack Method | Clean Acc. (Pre-Attack) | Impact on Clean Acc. | Attack Metric | Key Parameters | Full Results |
```

```
|:----------------|:-------------|:-----------------------|:--------------------|:------------
| Data Poisoning | Label Flipping | 82.93% | 65.78% | 65.78% (Degraded Acc.) | Flip Rate: 0.08, Target Class: 9 | [Details](../module2_attack_simulation/results/data_poisoning/label_flipping/label_flipping_report.md) |
| Backdoor | Static Patch | 82.93% | 69.40% | 3.00% (ASR) | Poison Frac.: 0.15, Target Class: 7, Patch Type: white_square | [Details](../module2_attack_simulation/results/backdoor/static_patch/static_patch_report.md) |
| Evasion | Pgd | 82.93% | 83.70% | 19.10% (Adv. Acc.) | Epsilon: 0.03, Num Iter: 50 | [Details](../module2_attack_simulation/results/evasion/pgd/pgd_report.md) |
| Evasion | Spsa | 82.93% | 82.40% | 29.20% (Adv. Acc.) | Epsilon: 0.03, Num Steps: 150, Delta: 0.01 | [Details](../module2_attack_simulation/results/evasion/spsa/spsa_report.md) |

**Note**: 'Clean Acc. (Pre-Attack)' represents the model's accuracy on clean data before any attack preparations. 'Impact on Clean Acc.' shows the model's accuracy on clean data *after* being subjected to the attack (e.g., trained with poisoned data, or backdoor injected). For Data Poisoning attacks, 'Attack Metric' displays the degraded accuracy of the model on clean inputs after poisoning. For Backdoor attacks, 'Attack Metric' displays the Attack Success Rate (ASR), indicating the percentage of adversarial samples (with trigger) successfully misclassified to the target class. For Evasion attacks, 'Attack Metric' displays the Adversarial Accuracy (Adv. Acc.) on perturbed inputs, where a lower value indicates a more successful attack.



## 5. Risk Analysis (Module 3)

This section summarizes the risk assessment performed on the simulated attacks. Each attack is evaluated based on its severity, probability, and visibility. A final risk score is computed to help prioritize mitigation strategies, followed by specific defense recommendations.


### 5.1 Risk Summary Table

| Attack | Type | Severity | Probability | Visibility | Risk Score | Report |
|:--------------|:--------------|:--------------|:--------------|:--------------|:--------------|
| Pgd | Evasion | 1.00 | 1.00 | 0.30 | 1.70 | [Details](../module2_attack_simulation/results/evasion/pgd/pgd_report.md) |
| Spsa | Evasion | 1.00 | 0.80 | 0.20 | 1.44 | [Details](../module2_attack_simulation/results/evasion/spsa/spsa_report.md) |
| Label Flipping | Data Poisoning | 0.57 | 1.00 | 0.62 | 0.79 | [Details](../module2_attack_simulation/results/data_poisoning/label_flipping/label_flipping_report.md) |
```

```
| Static Patch | Backdoor | 0.34 | 1.00 | 0.60 | 0.47 | [Details](../module2_attack_simulation/results/backdoor/static_patch/static_patch_report.md) |


### 5.2 Risk Matrix (Qualitative)

This matrix categorizes attacks based on their qualitative Severity and Probability levels.

| Severity \ Probability | Low | Medium | High |
|:-----------------------|:----------------------|:----------------------|:---------------------

| Low | - | - | - |
| Medium | - | - | Label Flipping, Static Patch |
| High | - | - | Pgd, Spsa |


### 5.3 Risk Ranking

Attacks ranked by their calculated Risk Score, from highest to lowest.

1. **Pgd**     Risk Score: 1.70     [Details](../module2_attack_simulation/results/evasion/pgd/pgd_report.md)
2. **Spsa**     Risk Score: 1.44     [Details](../module2_attack_simulation/results/evasion/spsa/spsa_report.md)
3. **Label Flipping**     Risk Score: 0.79     [Details](../module2_attack_simulation/results/data_poisoning/label_flipping/label_flipping_report.md)
4. **Static Patch**     Risk Score: 0.47     [Details](../module2_attack_simulation/results/backdoor/static_patch/static_patch_report.md)



### 5.4 Defense Recommendations

Based on the identified risks and threat profile, the following defense recommendations are provided:

- **Label Flipping**:
  - data_cleaning
  - per_class_monitoring
- **Pgd**:
  - adversarial_training
  - randomized_smoothing
- **Spsa**:
  - gradient_masking
  - jpeg_preprocessing
- **Static Patch**:
```

```
  - activation_clustering
  - spectral_signatures



### 5.5 Paths to Details

For more in-depth information about individual attacks, including raw metrics, attack visualizations, and specific parameters, please refer to the detailed reports linked in the 'Risk Summary Table' and 'Risk Ranking' sections above.

---
## 6. Defense Application (Module 4)
This section details the performance of the implemented defenses against the simulated attacks identified in the risk analysis. For each attack, the table shows the model's accuracy on clean data *before* and *after* defense, and the metric on malicious inputs *before* and *after* defense (ASR for backdoor, adversarial accuracy for evasion). Key defense parameters are also provided, along with a link to a detailed report.

| Attack Category | Attack Method | Defense Applied | Clean Acc. (Pre-Defense) | Metric on Malicious Inputs (Pre-Defense) | Clean Acc. (Post-Defense) | Metric on Malicious Inputs (Post-Defense) | Key Parameters | Link to Details |
|:---|:---|:---|:---|:---|:---|:---|:---|:---|
| Backdoor | Static Patch | Activation Clustering | 69.40% | ASR: 3.00% | 61.54% | ASR: 0.09% | num_clusters: 2 | [Details](../module4_defense_application/results/backdoor/static_patch/activation_clustering_report.md) |
| Backdoor | Static Patch | Spectral Signatures | 69.40% | ASR: 3.00% | 51.58% | ASR: 3.76% | threshold: 0.9 | [Details](../module4_defense_application/results/backdoor/static_patch/spectral_signatures_report.md) |
| Data Poisoning | Label Flipping | Data Cleaning | 65.78% | N/A | 68.37% | N/A | method: loss_filtering, threshold: 0.9 | [Details](../module4_defense_application/results/data_poisoning/label_flipping/data_cleaning_report.md) |
| Evasion | Pgd | Adversarial Training | 83.70% | Adv. Acc.: 19.10% | 76.55% | Adv. Acc.: 41.88% | attack_type: fgsm, epsilon: 0.03 | [Details](../
```

```
    module4_defense_application/results/evasion/pgd/adversarial_training_report.md)
                |
147 | Evasion           | Pgd             | Randomized Smoothing  | 83.70%
                        | Adv. Acc.: 19.10%                          | 37.20%
                         | Adv. Acc.: 19.88%                           | sigma:
    0.25                            | [Details](../module4_defense_application/
    results/evasion/pgd/randomized_smoothing_report.md)            |
148 | Evasion           | Spsa            | Gradient Masking      | 82.40%
                        | Adv. Acc.: 29.20%                          | 82.80%
                         | Adv. Acc.: 29.60%                           | strength:
    0.5                          | [Details](../module4_defense_application/results
    /evasion/spsa/gradient_masking_report.md)               |
149 | Evasion           | Spsa            | Jpeg Preprocessing    | 82.40%
                        | Adv. Acc.: 29.20%                          | 11.94%
                         | Adv. Acc.: 29.40%                           | quality:
    75                            | [Details](../module4_defense_application/
    results/evasion/spsa/jpeg_preprocessing_report.md)             |
150
151 **Note**:
152 - **Clean Acc. (Pre-Defense)**: Accuracy of the attacked or original model on clean
     data before applying defense. For data poisoning and backdoor, this is the
    compromised m o d e l s clean accuracy after poisoning/injection; for evasion,
    the original m o d e l s clean accuracy.
153 - **Metric on Malicious Inputs (Pre-Defense)**: For evasion,  Adv . Acc.  on
    adversarial examples before defense (lower means a stronger attack). For
    backdoor,  ASR  (Attack Success Rate) on triggered inputs before defense (
    higher means a more successful backdoor). Marked N/A for data poisoning.
154 - **Clean Acc. (Post-Defense)**: Accuracy on clean data after defense is applied;
    indicates how well clean performance is maintained or restored.
155 - **Metric on Malicious Inputs (Post-Defense)**: For evasion,  Adv . Acc.  on
    adversarial examples after defense (higher is better). For backdoor,  ASR
    on triggered inputs after defense (lower is better). Marked N/A for data
    poisoning.
156 - A lower ASR after defense indicates stronger mitigation of the backdoor; a higher
     Adv. Acc. after defense indicates stronger robustness against evasion.
157
158 ### 6.1 Applied Defenses and their Purposes
159 The following defenses were applied and evaluated to mitigate the identified risks:
160
161 * **Activation Clustering**: A defense aimed at detecting and neutralizing
    backdoors in models. It works by clustering intermediate layer activations of
    the model to identify and isolate training samples containing the malicious
    trigger, allowing for their removal.
162 * **Adversarial Training**: One of the most common and effective defenses against
    evasion attacks. It involves augmenting the training dataset with adversarial
    examples (generated by the attack itself) and retraining the model. This
    improves the model's robustness, making it more resistant to future adversarial
     perturbations.
```

```
* **Data Cleaning**: A general approach to remove corrupted, mislabeled, or outlier samples from the training dataset. It aims to improve the overall quality and integrity of the data, thereby making the model more robust to various forms of data-based attacks, including poisoning.
* **Gradient Masking**: This defense aims to obscure or modify the gradients seen by an attacker, making it harder for gradient-based adversarial attacks to succeed. It can involve various techniques like non-differentiable transformations or adding noise to gradients.
* **Jpeg Preprocessing**: A simple defense that applies JPEG compression to inputs before feeding them to the model. The compression process can flatten out small adversarial perturbations, making the adversarial examples less effective against the model.
* **Randomized Smoothing**: A certified defense that provides provable robustness guarantees against adversarial attacks. It works by adding random noise to inputs during inference and then classifying based on the aggregated predictions, making it difficult for an attacker to craft effective adversarial examples.
* **Spectral Signatures**: A backdoor detection technique that analyzes the spectral properties of the hidden layer activations. It identifies anomalous patterns indicative of a backdoor trigger embedded in the training data, allowing for the isolation and mitigation of poisoned samples.

---
## 7. Defense Evaluation (Module 5)
This section presents the evaluation of applied defenses, summarizing their mitigation effectiveness, impact on clean accuracy, computational/resource cost, and overall final score. We also highlight the top-performing defenses for each attack method and discuss key observations.

### 7.1 Summary Table of Defense Evaluation Scores

| Attack Category   | Attack Method   | Defense               |   Mitigation |   CAD |   Cost |   Final Score |
|:------------------|:----------------|:----------------------|-------------:|------:|-------:|--------------
| Backdoor          | Static Patch    | Activation Clustering |         0.97 |   0.887 |    0.3 |         0.926 |
| Backdoor          | Static Patch    | Spectral Signatures   |         0    |   0.743 |    0.5 |         0.142 |
| Data Poisoning    | Label Flipping  | Data Cleaning         |         0.151 |   1.039 |    0.2 |         0.322 |
| Evasion           | Pgd             | Adversarial Training  |         0.282 |   0.923 |    0.8 |         0.38  |
| Evasion           | Pgd             | Randomized Smoothing  |         0.01  |   0.449 |    0.5 |         0.093 |
| Evasion           | Spsa            | Gradient Masking      |         0.006 |   0.998 |    0.4 |         0.196 |
| Evasion           | Spsa            | Jpeg Preprocessing    |         0.003 |   0.144 |    0.1 |         0.031 |
```

```
184
185 ### 7.2 Top-Performing Defenses
186
187 - **Backdoor / Static Patch**: Top defense is **Activation Clustering** (Mitigation
        : 0.970, CAD: 0.887, Cost: 0.300, Final Score: 0.926).
188 - **Data Poisoning / Label Flipping**: Top defense is **Data Cleaning** (Mitigation
        : 0.151, CAD: 1.039, Cost: 0.200, Final Score: 0.322).
189 - **Evasion / Pgd**: Top defense is **Adversarial Training** (Mitigation: 0.282,
        CAD: 0.923, Cost: 0.800, Final Score: 0.380).
190 - **Evasion / Spsa**: Top defense is **Gradient Masking** (Mitigation: 0.006, CAD:
        0.998, Cost: 0.400, Final Score: 0.196).
191
192 ### 7.3 Observations and Recommendations
193
194 Based on the evaluation scores above, consider the following:
195
196 **Detailed per-attack-method rankings:**
197 **Backdoor / Static Patch**:
198 - Activation Clustering: Final Score 0.926 (Mitigation 0.970, CAD 0.887, Cost
        0.300)
199 - Spectral Signatures: Final Score 0.142 (Mitigation 0.000, CAD 0.743, Cost 0.500)
200
201 **Data Poisoning / Label Flipping**:
202 - Data Cleaning: Final Score 0.322 (Mitigation 0.151, CAD 1.039, Cost 0.200)
203
204 **Evasion / Pgd**:
205 - Adversarial Training: Final Score 0.380 (Mitigation 0.282, CAD 0.923, Cost 0.800)
206 - Randomized Smoothing: Final Score 0.093 (Mitigation 0.010, CAD 0.449, Cost 0.500)
207
208 **Evasion / Spsa**:
209 - Gradient Masking: Final Score 0.196 (Mitigation 0.006, CAD 0.998, Cost 0.400)
210 - Jpeg Preprocessing: Final Score 0.031 (Mitigation 0.003, CAD 0.144, Cost 0.100)
            marginal improvement; likely not worth deploying alone.
211
212 **Overall Recommendation:**
213 - **Backdoor / Static Patch**: Top defense is **Activation Clustering** (Final
        Score 0.926)     Recommended.
214 - **Data Poisoning / Label Flipping**: Top defense is **Data Cleaning** (Final
        Score 0.322)     Recommended.
215 - **Evasion / Pgd**: Top defense is **Adversarial Training** (Final Score 0.380)
            Recommended.
216 - **Evasion / Spsa**: Top defense is **Gradient Masking** (Final Score 0.196)
        Recommended.
217
218 For more details, refer to the full defense evaluation report: [Details](../
        module5_defense_evaluation/results/defense_evaluation_report.md).
219
220 ---
221 ## 8. Conclusions and Executive Summary
```

```
222
223 **Highest-Risk Attack:** Pgd (Risk Score: 1.700).
224 - Severity: 1.000, Probability: 1.000, Visibility: 0.300.
225 **Also high risk:** Spsa (1.440), Label Flipping (0.787).
226
227 **Most Effective Defenses Identified:**
228 - Against **Static Patch**, top defense: **Activation Clustering** (Final Score:
      0.926).
229 - Against **Label Flipping**, top defense: **Data Cleaning** (Final Score: 0.322).
230 - Against **Pgd**, top defense: **Adversarial Training** (Final Score: 0.380).
231 - Against **Spsa**, top defense: **Gradient Masking** (Final Score: 0.196).
232
233 **Overall Security Posture:**
234 - Pgd identified as highest risk. Effective defenses identified for most attacks,
      except some evasion methods.
235
236 **Practical Recommendations:**
237 - Prioritize deploying **Activation Clustering** against Static Patch.
238 - Prioritize deploying **Data Cleaning** against Label Flipping.
239 - Prioritize deploying **Adversarial Training** against Pgd.
240 - Prioritize deploying **Gradient Masking** against Spsa.
241
242 ---
243 ## 9. Recommendations for Continuous Monitoring and Post-Deployment
244
245 ### 9.1 Monitoring Metrics
246 - **Input Distribution Monitoring**: Continuously track statistics of incoming data
       (e.g., feature distributions, class frequencies). Unexpected shifts may signal
      data drift or adversarial attempts.
247 - **Model Performance Metrics**: Monitor live accuracy or proxy metrics on clean-
      like validation streams if available. Sudden drops could indicate emerging
      attacks or data issues.
248 - **Confidence and Uncertainty**: Log model confidence scores and uncertainty
      metrics (e.g., softmax entropy). A rise in low-confidence predictions or
      abnormal confidence patterns can hint at adversarial inputs.
249 - **Error Rates per Class**: Track per-class error rates over time. Spikes in
      errors for specific classes may indicate targeted data poisoning or evolving
      distribution shifts.
250 - **Resource Usage and Latency**: Monitor inference latency and resource
      consumption, especially if defenses (e.g., input preprocessing) are in place.
      Degradation may affect user experience and could be exploited.
251
252 ### 9.2 Periodic Re-assessment
253 - **Scheduled Security Audits**: Automate rerunning Modules 2  5   on updated data
      or model versions at regular intervals (e.g., quarterly or upon major model
      updates).
254 - **Retraining with Fresh Data**: If new data is collected over time, include it in
       retraining pipelines with relevant attack/defense simulations to ensure up-to-
      date robustness.
```

```
- **Threat Landscape Updates**: Stay informed about new attack methods; incorporate new simulations and defenses into the framework as they emerge.
- **Regression Testing**: After model updates or defense adjustments, re-evaluate known attack scenarios to ensure no regressions in vulnerability.

### 9.3 Alerting and Thresholds
- **Define Alert Conditions**: Establish thresholds for monitored metrics (e.g., sudden shift in input distribution, drop in accuracy beyond X%, unusual rise in low-confidence predictions).
- **Automated Alerts**: Connect monitoring to alerting systems (e.g., email, Slack) to notify stakeholders when thresholds are crossed.
- **Anomaly Detection on Incoming Requests**: Deploy runtime anomaly detection to flag suspicious inputs (e.g., out-of-distribution or adversarial patterns).
- **Logging and Auditing**: Maintain detailed logs of input features, predictions, confidence, and any preprocessing steps, to facilitate forensic analysis after an alert.

### 9.4 Data Drift and Concept Drift
- **Drift Detection Techniques**: Use statistical tests or drift-detection algorithms (e.g., population stability index, KS test) to identify significant changes in input feature distributions.
- **Model Retraining Triggers**: Define criteria for retraining when drift is detected (e.g., sustained drift over time or performance degradation beyond threshold).
- **Adversarial Drift Monitoring**: Pay attention to shifts that may be induced by adversarial behavior; correlate drift alerts with security incidents.

### 9.5 Model Versioning and Rollback Plans
- **Version Control for Models**: Tag and store model artifacts (weights, configurations) with version identifiers. Ensure easy retrieval of previous safe versions.
- **Canary Deployments**: Roll out new model versions to a subset of traffic first; monitor metrics closely before full deployment.
- **Rollback Procedures**: Define automated or manual rollback processes if new vulnerabilities or performance regressions are detected.
- **A/B Testing Isolation**: When testing different model versions, isolate traffic to prevent cross-contamination of potential attacks.

### 9.6 Security Incident Response
- **Incident Response Plan**: Document steps to take when an attack or anomaly is detected (e.g., isolate affected services, trigger deeper forensic analysis).
- **Containment Strategies**: If an ongoing attack is detected (e.g., data poisoning detected in training pipeline), halt training or deployment until issue is resolved.
- **Remediation Actions**: Procedures for retraining or patching the model, applying additional defenses, or updating preprocessing pipelines.
- **Post-Incident Review**: After an incident, analyze root causes, update threat model assumptions, and refine monitoring and defense strategies accordingly.
```

```
- **Stakeholder Communication**: Establish clear communication channels and
    responsibilities for notifying relevant teams (e.g., security, ML engineering,
    product) during incidents.

### 9.7 Integration into CI/CD Pipelines
- **Automated Security Checks**: Incorporate automated runs of attack/defense
    simulations (Modules 2  5  ) into CI pipelines triggered by code or data changes
    .
- **Gatekeeping Deployments**: Block deployments if security evaluation metrics
    fall below predefined thresholds (e.g., risk score above threshold, defense
    evaluation final score below threshold).
- **Continuous Reporting**: Generate and archive periodic security reports; notify
    stakeholders of changes in security posture.
```

Listing B.3: Contents of `final_report.md` generated by Module 6