Thesis submitted to the

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA, ALGERIA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

in Partial Fulfillment of the Requirements for the Degree of:

Master's in Computer Science

Specialization: Business Intelligence and Optimization

By

Cheyma Dilmi, Warda Loukriz

Entitled

Detection of Poisoned Data Using Genetic Algorithm : Case Study on Badnet Attack in Image Classification Models

Under the supervision of

Hemmak Allaoua

Under the co-supervision of

Wafa Bouras

Jury Members

Said Hamani	University of M'sila	President
Hemmak Allaoua	University of M'sila	Supervisor
Wafa Bouras	University of M'sila	Co-supervisor
Tahar Mehenni	University of M'sila	Examiner

June, 2026

الملخص

أدى الاعتماد المتزايد على نماذج التعلم العميق، خاصة في مهام تصنيف الصور، إلى ظهور تحديات كبيرة تتعلق بالأمن والموثوقية. ومن بين أخطر هذه التهديدات هجمات تسميم البيانات، لا سيما هجمات الأبواب الخلفية مثل BadNets، حيث يتم إدخال أنماط خبيثة داخل بيانات التدريب بهدف التلاعب بسلوك النموذج. وتُعد هذه الهجمات صعبة الاكتشاف بسبب طبيعتها الخفية وتأثيرها المحدود على دقة النموذج أثناء التقييم التقليدي.

تقترح هذه الأطروحة إطارًا جديدًا للكشف يعتمد على الخوارزميات الجينية لتحديد البيانات المسمومة ضمن مجموعات بيانات الصور. تعتمد هذه المقاربة على تقنيات التحسين التطوري لاستكشاف فضاء البيانات والتمييز بين العينات السليمة والمصابة، مع تصميم دالة ملاءمة مخصصة لتحسين دقة عملية الكشف.

أظهرت النتائج التجريبية أن الطريقة المقترحة تحقق أداءً واعدًا في اكتشاف البيانات المسمومة وعزل العينات الخبيثة، مما يبرز أهمية الخوارزميات التطورية في تعزيز أمن أنظمة التعلم العميق.

الكلمات المفتاحية: التعلم العميق، تسميم البيانات، BadNets، الخوارزميات الجينية، تصنيف الصور.

Résumé

La dépendance croissante aux modèles d'apprentissage profond, notamment dans les tâches de classification d'images, a soulevé des préoccupations majeures concernant leur sécurité et leur robustesse. Parmi les menaces les plus graves figurent les attaques par empoisonnement de données, en particulier les attaques par porte dérobée telles que BadNets, dans lesquelles des motifs malveillants sont intégrés aux données d'entraînement afin de manipuler le comportement du modèle. Ces attaques sont difficiles à détecter en raison de leur nature furtive et de leur impact limité sur les performances globales du modèle lors des évaluations classiques.

Cette thèse propose un cadre de détection basé sur les Algorithmes Génétiques (AG) pour identifier les données empoisonnées au sein de jeux de données d'images. L'approche repose sur des techniques d'optimisation évolutionnaire pour explorer l'espace des données et distinguer les échantillons sains des échantillons compromis, soutenue par une fonction d'adaptation personnalisée visant à améliorer la précision de la détection.

Les résultats expérimentaux démontrent que la méthode proposée atteint des performances prometteuses dans la détection des données empoisonnées et l'isolation des échantillons malveillants, mettant en évidence l'efficacité des algorithmes évolutionnaires pour renforcer la sécurité des systèmes d'apprentissage profond.

Mots-clés : Apprentissage profond, Empoisonnement de données, BadNets, Algorithme génétique, Classification d'images.

Abstract

The increasing reliance on deep learning models, particularly in image classification tasks, has raised critical concerns regarding their security and robustness. Among the most serious threats are data poisoning attacks, especially backdoor attacks such as BadNets, where malicious patterns are embedded into training data to manipulate model behavior. These attacks are difficult to detect due to their stealthy nature and minimal impact on overall model performance.

This thesis proposes a detection framework based on Genetic Algorithms (GAs) to identify poisoned data within image datasets. The approach uses evolutionary optimization techniques to explore the data space and distinguish between clean and compromised samples, supported by a customized fitness function to improve detection accuracy.

Experimental results demonstrate that the proposed method achieves promising performance in detecting poisoned data and isolating malicious samples, highlighting the effectiveness of evolutionary algorithms in enhancing the security of deep learning systems.
Keywords: Deep Learning, Data Poisoning, BadNets, Genetic Algorithm, Image Classification.

Dedication

إلى نفسي

إلى أبي الغالي إني لا أملك من نفسي شيئا لولا جهدك و تعبك

إلى اليد الحانية التي امتدت لتعلمني الكتابة و القراءة و رعتنا حتى صرنا كبار أمي الحبيبة

أتمنى أن أكون بهذا الانجاز قد عوضتكما و لو بالقليل .

و إلى الذين كانوا مطرا أغاث خَطَ الطريق فأورق، وعَضدا شددت به أزري فِي لواء الكد إخوتي .

لكل الذين جعلوا منَ المسير المضني رحلة أنس .

و إلى لقاء قريب نكون فيه حيث تمنينا .

شيماء ديلمي و لقريز وردة

Acknowledgements

شكر و تقدير

نشكر الله العلي القدير الذي أنعم علينا بنعمة العقل والدين و الذي وفقنا لإكمال بحثنا المتواضع, كما يسرني أن أتقدم بجزيل الشكر لوالديَّ اللذين قدما لي يد العون و سهرا على تربيتي و تعليمي منذ الصغر ، و أتوجه بالشكر لكل من ساهم في تدريسي و تعليمي أي معلومة نافعة من أساتذة و دكاترة سواء في المدرسة أو الجامعة و لأولئك المخلصين الذين لم يألوا جهداً في سبيل مساعدتنا الأستاذة وفاء والأستاذ هماك فجزاهما الله عنا كل خير كما نتقدم بخالص الشكر إلى الأساتذة أعضاء لجنة المناقشة على قبولهم مناقشة هذا العمل و على ما سيبذلونه من وقت و جهد في قراءته و تقييمه.

Contents

List of Tables

List of Figures

General Introduction

Deep learning has undergone considerable transformation in recent years and has become a cornerstone technology for intelligent visual assessment across numerous critical domains, including autonomous driving, medical imaging, and security surveillance. As image classification models grow in complexity and are deployed in increasingly sensitive environments, ensuring their safety and dependability has become a fundamental concern. Among the most serious threats targeting these systems are backdoor attacks, a class of data poisoning attacks that embed hidden malicious behaviors into a model during training, causing it to misclassify any input carrying a specific trigger pattern while behaving normally on clean data.

The BadNet attack represents the foundational instance of this threat. By injecting a small number of poisoned training samples, each stamped with a localized pixel trigger, an adversary can silently compromise a deep learning model without raising any performance alarms during standard evaluation. The stealthy nature of such attacks renders conventional detection methods largely ineffective, as poisoned samples are carefully crafted to be statistically indistinguishable from legitimate training data.

This thesis addresses this challenge by proposing a novel detection framework based on Genetic Algorithms (GAs). Rather than relying on statistical outlier detection or model internals, the proposed approach treats trigger localization as an evolutionary optimization problem. The framework searches for the precise spatial region of the backdoor trigger by applying erasure patches to reference images and observing changes in the model's prediction behavior operating entirely under a black-box assumption with no access to model weights or gradients.

Core Problems Addressed:

- The hidden and stealthy nature of backdoor triggers embedded in visual training data.
- The limitations of traditional, fixed detection methods that assume poisoned samples appear as statistical outliers.
- The absence of adaptive, search-driven strategies capable of localizing triggers without prior knowledge of their shape, size, or location.

Research Objectives:

1. To develop a novel Genetic Algorithm framework capable of examining image datasets and identifying backdoor triggers embedded in Convolutional Neural Networks (CNNs).
2. To evaluate the performance of the proposed GA framework in distinguishing clean data from poisoned data in a BadNet backdoor attack context.
3. To design a fitness function that accurately guides the evolutionary search toward the true trigger region while preserving the integrity of clean samples.

Research Questions:

- How can a genetic algorithm be designed and configured to detect poisoned samples within image datasets used to train convolutional neural networks?
- To what extent can the proposed GA framework reliably distinguish clean data from poisoned data under a BadNet backdoor attack?
- What role does the fitness function formulation play in improving the accuracy and robustness of backdoor trigger detection?

Document Organization:
This thesis is organized into four chapters:

- **Chapter 1** introduces the theoretical foundations of artificial intelligence, machine learning, and deep learning, and presents the security threats targeting these systems, with particular focus on data poisoning and backdoor attacks, including a detailed case study on the BadNet attack and an overview of Genetic Algorithms.
- **Chapter 2** provides a comprehensive review of the related literature on data poisoning attacks, existing detection and defense mechanisms, and the application of Genetic Algorithms in security contexts. It identifies the key research gaps that motivate this work.
- **Chapter 3** presents the proposed methodology and system design. It formalizes the problem, describes the chromosome encoding, fitness function, evolutionary operators, and system architecture, and presents the two-phase detection pipeline.
- **Chapter 4** reports the experimental evaluation and comparative analysis of the proposed framework on the CIFAR-10 dataset under BadNet attack conditions, and benchmarks the results against established detection methods.

Chapter 1

Theoretical Foundations and Deep Learning

1.1 Introduction

As artificial intelligence and deep learning systems become increasingly widespread, ensuring the integrity and reliability of training data has become a critical concern. While these models achieve high performance in tasks such as image classification, they remain vulnerable to security threats, particularly data poisoning and backdoor attacks.

This chapter presents the fundamental concepts underlying this work, including artificial intelligence, machine learning, and data quality. It also introduces deep learning and its applications in computer vision, followed by an overview of security challenges in these systems. Finally, it discusses data poisoning attacks, with a focus on BadNets, and outlines the research framework and key concepts related to genetic algorithms used in this study.

1.2 Artificial Intelligence and ML Foundations

1.2.1 Artificial Intelligence Definition

Artificial intelligence (AI) might not have a universally accepted definition, yet an individual discussing its past should certainly have a specific understanding in mind. In the context of this study , artificial intelligence refers to the effort focused on developing intelligent machines, and intelligence is the characteristic that allows an entity to operate effectively and with the ability to anticipate events in its surroundings [1].

1.2.2 Machine Learning (ML) :

Machine learning is an analytic approach that identifies patterns in incoming data to automatically perform tasks without continuous human intervention. As a core component of

Artificial Intelligence, it serves as a foundational platform for both governmental and industrial applications [2]. The primary objective of machine learning is to accurately predict a target outcome l , which belongs to the set Y, by minimizing a loss function $L(f(x), l)$. This process aims to approximate a function :

$$f : X \to Y \tag{1.1}$$

that maps input features X to outputs Y. Typically, model performance improves with larger datasets, as the system can capture more complex patterns. By learning directly from data, machines can solve specific problems without relying on extensive manual programming. Broadly, machine learning methods are categorized into four main types, which will be discussed in the following sections [3].

1.2.3 Core Components : R+E+O

According to [4], any machine learning algorithm is essentially a combination of three key components :

1. **Representation :** This involves choosing a formal language for the classifier, which effectively defines the "hypothesis space" the set of solutions the learner can possibly discover.

2. **Evaluation :** An evaluation function, often called an objective or scoring function, is needed to distinguish good classifiers from bad ones based on their performance.

3. **Optimization :** Finally, a search method is required to navigate through the available classifiers to find the one that achieves the highest score.

Since these three stages rely entirely on the integrity of the training data, any "poisoning" of that data will naturally corrupt the final results. This makes the detection of such malicious inputs a vital necessity for the security of the learning process.

1.2.4 The Machine Learning Lifecycle (ML Lifecycle)

The machine learning lifecycle is a complex iterative process that aims to develop safe and reliable AI models. This cycle consists of four essential integrated stages, and the first three stages are collectively called the machine learning workflow [5] :

1. **Data Management :** This foundational step involves collection, preprocessing, augmentation, and analysis, addressing vulnerabilities where attackers may introduce malicious data.

2. **Model Learning :** In this stage, data is utilized for model synthesis through algorithm selection, training, and hyperparameter tuning. If training data is compromised, security vulnerabilities can be encoded into the model.

3. **Model Verification :** This stage is the process of demonstrating a model's generalization and accuracy on unseen data, consisting of test-based verification that evaluates performance using a verification data set to compute generalization error, and formal verification, which uses mathematical proofs to confirm adherence to specified security properties.

4. **Model Deployment :** The final stage involves integrating the verified model into operational systems, with ongoing monitoring and updates as needed.

1.2.5 Classification of Machine Learning Algorithms

Machine learning algorithms are divided into four main categories according to the type of data that is accessible and the way the system engages with its surroundings throughout the training phase [6]:

1. **Supervised Learning (Task-Driven) :** This approach connects labeled inputs to outputs for tasks like regression and classification. Examples include analyzing sentiments in tweets or product feedback and predicting category labels.

2. **Unsupervised Learning (Data-Driven) :** it uncovers latent patterns in unstructured data autonomously through processing. Key activities include clustering, estimating density, reducing dimensionality, and identifying anomalies.

3. **Semi-supervised Learning (Hybrid) :** combines a small number of costly labeled examples with a large pool of unlabeled data, enhancing prediction accuracy. This method is applied in fraud detection and machine translation.

4. **Reinforcement Learning (Environment-Driven) :** This technique enhances an agent's actions by experimenting and receiving rewards or penalties. It is utilized in advanced systems such as robotics, autonomous vehicles, and supply chain management.

1.3 Data Quality and Integrity in ML :

As stated in [7], the trustworthiness of machine learning systems is essentially connected to the qualities of the information they utilize. This can be divided into two main aspects:

1.3.1 Definitions of Data Quality and Integrity

- Data Quality pertains to the general state of a dataset regarding its accuracy, consistency, completeness, and reliability. Data of superior quality guarantees that machine learning algorithms are provided with significant and credible inputs, resulting in enhanced forecasting and improved decision-making.
- Data Integrity refers to the guarantee that information stays precise, consistent, and unchanged during its entire existence. It encompasses upholding accuracy, stopping unauthorized changes, and ensuring the ability to trace data

1.3.2 Classification of Real-World Data Types :

Within the realm of machine learning, the presence and qualities of data are fundamental in building strong models and implementing systems that rely on data. Data from the real world presents itself in multiple formats, mainly divided into structured, semi-structured, and unstructured categories, in addition to metadata [6] :

1. Structured Data: This type of data is meticulously arranged according to a specific tabular format, commonly found in relational databases, dates, and numerical records.
2. Semi-Structured Data: This form of data does not fit into strict table formats but includes organizational indicators or tags for the purpose of analysis, as seen in JSON and XML documents.
3. Unstructured Data: Data lacking a specific format is considered the most challenging to analyze. Examples of unstructured data include digital images (the primary subject of this study), videos, and raw text, all of which require sophisticated deep learning models for comprehension.
4. Metadata: Referred to as "data about data," it delivers crucial context surrounding the main dataset, including details such as file size, creator, and timestamps of creation.

Although various fields employ distinct data formats, the disorganized characteristics of image databases create considerable weaknesses. This complexity in structure emphasizes the importance of maintaining data quality and integrity to protect against hostile alterations.

1.3.3 Key Dimensions of Data Quality(Focus on Image Data):

The effectiveness and suitability of data for training are measured across several key dimensions that determine its quality as discussed in [7] , the most important of which are :

- **Accuracy :** Verifying that the details are true, precise, and represent actual values without mistakes.
- **Completeness :** Completeness: Checking that all required data entries are present and that no vital information is missing.
- **Consistency :** Consistency: Making sure there is uniformity across datasets, preventing any inconsistencies or contradictory values.
- **Timeliness :** Ensuring that the information is up-to-date and available when needed for assessment or model training.
- **Validity :**Confirming that the data complies with specified standards, formats, and requirements (e.g., proper email addresses, accurate date formats).
- **Uniqueness :** Preventing duplicates and verifying that each piece of data is individual.

1.3.4 Importance of Data Quality and Integrity in Machine Learning :

Primary factors that highlight the importance of data quality and integrity in machine learning encompass [7] :

1. Enhanced Model Accuracy: Consistent and reliable data leads to models that effectively generalize and yield precise predictions.
2. Decrease in bias and errors: Subpar data quality may introduce biases, resulting in unjust or discriminatory results.
3. Optimal Use of Resources: Handling low-quality data can cause the inefficient use of computational power and prolonged training durations.
4. Adherence to Regulatory and Compliance Standards: Numerous sectors adopt stringent data governance regulations that necessitate organizations upholding superior standards of data integrity.

1.4 Deep Learning and Computer Vision Foundations

1.4.1 Deep Learning (DL) Definition

Deep learning (DL) is a subfield of machine learning that automatically learns high-level representations from raw data through multiple computational layers. By employing artificial neural networks, particularly Convolutional Neural Networks (CNNs), deep learning overcomes many limitations of traditional machine learning methods, enabling efficient processing and classification of large-scale unstructured data [8].

1.4.2 Classification of Machine Learning Paradigms

According to [9], machine learning methods are commonly categorized into three main paradigms based on the availability of labeled data:

- **Supervised Learning:** This approach relies on labeled datasets, where each input is associated with a corresponding output, typically represented as $(x, y) \sim \mathcal{D}$. It is widely used for classification and regression tasks, where models learn by minimizing a predefined loss function. However, it requires large amounts of accurately annotated data, which can be costly and time-consuming to obtain.
- **Semi-Supervised Learning:** This method combines a limited amount of labeled data with a larger set of unlabeled data, leveraging both to improve model performance. It offers a practical compromise between supervised and unsupervised learning, reducing annotation costs while maintaining good predictive accuracy.
- **Unsupervised Learning:** This approach operates without labeled data and aims to discover hidden patterns or structures within the data. Common techniques include clustering and dimensionality reduction, often implemented using models such as autoencoders and generative adversarial networks.

1.4.3 Artificial Neural Networks (ANN): The Building Blocks

Artificial neural networks are computational models inspired by the human brain, composed of interconnected processing units (neurons). These networks learn from data by adjusting the weights of connections between neurons, enabling them to capture complex patterns and relationships. The most widely used type of artificial neural network consists of three groups of units or layers: an "input" layer that is linked to a "hidden" layer and then to an "output" layer.There are three layers that compose the Architecture of an Artificial Neural Network [10] :

1. **The Input Layer :**

   The input layer of a neural network comprises a group of artificial input neurons. These neurons transmit data from the primary neuron layers to the network for further processing. The input layer serves as the starting point for the neural network's operations.

2. **The Hidden Layer :**

   The hidden layer of the artificial neural network is formed by layers for input and output, where the inputs and outputs of the artificial neurons are adjusted based on the

quantity of inputs received.

3. **The Output Layer :**

   The output layer is the final layer of neurons in an artificial neural network, delivering specific results within the programming context. Neurons in the output layer act as the ultimate "performers" and can be constructed and managed in distinct ways.

1.4.4 Deep Learning Architectures

This section covers the most well-known varieties of deep learning frameworks [10], which consist of :

Recursive neural networks (RvNNs)

Created to handle data that is organized in a hierarchical or tree format, including graphs and sentences. Although they are very effective in natural language processing for understanding syntax and structure, they are fundamentally different from architectures that focus on extracting spatial features from visual data sets.

Feed Forward Neural Network (FNN)

Stand as the essential category of artificial neural networks, mainly employed for elementary classification and regression assignments. Within this framework, data progresses solely in one direction advancing from the input layer, passing through the hidden layers, and arriving directly at the output layer without any cycles or feedback mechanisms. A fundamental element of this design is the perceptron, which acts as a linear binary classifier and represents the foundational model for supervised learning contexts.

Recurrent Neural Networks (RNN)

Represent a sophisticated category of networks that include connections from the output back to the hidden layers, thus establishing a feedback loop or a direct memory cycle. This time-based feature enables the network to handle continuous sequences of data. While mainly used in natural language processing, these networks can also be applied to certain tasks involving time-series image processing.

Convolutional Neural Networks (CNN)

- In the realm of deep learning, Convolutional Neural Networks (CNNs) stand out as the most prominent and commonly employed algorithms for applications including computer vision, speech analysis, and facial recognition. Their key benefit compared to

conventional models lies in their capacity to autonomously detect and discern important features without the need for human intervention. Modeled after the visual cortex, CNNs handle two-dimensional information like images by utilizing shared weights and localized connections rather than relying on fully interconnected networks. This methodology significantly lowers the quantity of training parameters, easing the learning process and enhancing the speed of the network.

- Within this, the input x for each layer is structured in three dimensions: height, width, and depth, represented as $m \times m \times r$, with height (m) being equal to width. The depth, often referred to as the number of channels, indicates the color information; for instance, in an RGB image, the depth (r) is three. Each convolutional layer consists of several kernels (filters), denoted as k, which also have three dimensions ($n \times n \times q$), similar to the dimensions of the input image.[11]

1. **Basic Architecture :**

   The primary layers that characterize CNNs consist of various components that change and gather insights from the input data [12].

   - **Convolutional Layers :**

     Utilizing a technique called convolution, these layers apply filters to the input data to pull out features, enabling the identification of edge boundaries, texture patterns, and shapes.

   - **Activation function :**

     These functions add non-linear characteristics to modify the outputs generated by convolution, allowing the network to recognize intricate patterns. Although there are choices such as sigmoid and tanh, ReLU and its adaptations are favored to address the vanishing gradient issue, along with more recent functions like Mish.

   - **Batch normalization :**

     This method tackles the internal shift in covariance, which refers to the change in the distribution of hidden unit values that hampers the training process. By adjusting the feature-map values to have a mean of zero and a variance of one, it normalizes their distribution, stabilizes calculations, facilitates the gradient flow, and enhances the network's ability to generalize.

- **Pooling Layers :**

  These layers diminish the spatial dimensions of the data, lowering the processing demands while providing the system with translation invariance. Average pooling and max pooling are the two common pooling techniques utilized in CNN processing.

- **Dropout :**

  This acts as a regularization mechanism to avert the issue of overfitting that arises from the mutual adjustment of various network connections. By randomly omitting some units or links during the training phase with a defined likelihood, it generates streamlined structures that resemble a more generalized and effective final network.

- **Fully Connected Layers :**

  The network evaluates the data through its fully connected layers, where it formulates the final predictions based on the features it has extracted illustrates a typical CNN framework, displaying convolutional, pooling, and fully connected layers in a sequential manner.

2. **CNN based image classification :**

   Convolutional Neural Networks (CNNs) find extensive application in image classification across different areas. One of the main uses is in healthcare, particularly in the identification of cancer through histopathological images, including the detection of breast cancer. In these scenarios, CNN frameworks exhibit excellent performance, which is frequently improved by methods like data augmentation to manage issues related to class imbalance. Furthermore, CNN designs are effectively employed in transport systems, such as the classification of traffic signs, attaining impressive recognition rates on standardized datasets [12].

1.5 Security in Image-Based Deep Learning Systems :

Safeguarding modern autonomous systems and Vision-related activities depending on Deep Learning (DL) frameworks, especially Convolutional Neural Networks (CNNs), is now of top priority. Though these systems can find sophisticated patterns inside data, their reliance on the caliber of training data generates significant flaws. Adversarial Machine Learning

explores this important relationship between great model performance and security, investigating how intentional changes like Data Poisoning can erode a model's reliability. Understanding these hazards is a vital prerequisite for preserving the stability and dependability of artificial intelligence systems in real-world applications as well as an intellectual endeavor.

1.5.1 Adversarial Machine Learning Overview

Adversarial machine learning exists where machine learning meets security, focusing on recognizing, utilizing, and protecting against the vulnerabilities present in machine learning models. Since the groundbreaking findings by Szegedy [13]and colleagues, there has been a significant surge in research within this domain, investigating not only the creation of adversarial examples but also the wider security ramifications of utilizing machine learning models in hostile settings.The threats from adversarial attacks are now beyond mere theoretical discussions; they represent actual dangers in real-world applications. For instance, adversarial alterations can lead self-driving cars to incorrectly interpret road signs, mislead medical imaging diagnostic tools, or enable financial fraud detection systems to be bypassed.

1.5.2 Classifcation of Adversarial Attacks:

To create a thorough security evaluation of Machine Learning systems, it is essential to classify malicious threats depending on the point of intervention and the adversary's final objectives. As outlined in the NIST AI 100-2e2023 framework [14], these attacks are organized across these specific aspects:

1. **Training-time attacks :**

   The weakness of a machine learning model greatly relies on the timing of the adversary's involvement. The field of adversarial machine learning distinguishes two key phases:

   - **Training Phase (Poisoning Attacks):**

     This reflects a direct, causative effect whereby the attacker influences the learning process. An adversarial party adds or changes components of the training data in a Data Poisoning attack to methodically reduce the model's performance in subsequent phases.

- **Deployment Stage (Evasion Attacks):**

  This is an example of an exploratory influence where the trained model stays unaltered. The opponent instead creates adversarial examples by small, focused changes to the inputs in order to trick the model during inference.

2. **Attacker Goals and Objectives :**

The objectives of the attacker can be grouped into three distinct categories depending on the three main types of security violations examined when assessing a system's protection (which are availability, integrity, and confidentiality): disruption of availability, violations of integrity, and threats to privacy. In this framework, the success of an adversary indicates the achievement of one or more of these objectives.

- **Availability Breakdown :**

  In this form of attack, the opponent seeks to indiscriminately reduce or entirely disrupt the overall efficiency of the machine learning model when it is deployed. This can be done through techniques such as data poisoning, model poisoning, or energy-latency attacks

- **Integrity Violations :**

  This type of attack aims at undermining the trustworthiness of the outputs generated by the machine learning model, leading to the creation of inaccurate predictions (misclassifications) while remaining undetectable to humans. This is accomplished through evasion attacks at the point of deployment, or via targeted and backdoor poisoning attacks during the training phase.

- **Privacy Compromise :**

  This aspect concentrates on the unauthorized retrieval of information. It is categorized into Data Privacy attacks (including data reconstruction and membership inference aimed at acquiring knowledge about the training data) and Model Privacy attacks (like model extraction intended to obtain confidential information regarding the model itself).

3. **Attacker Capabilities**

In order to reach their harmful goals, an opponent might utilize different security Capabilities based on how much access they have to the Machine Learning (ML) system.

These features are organized into six unique categories:

- **TRAINING DATA CONTROL:**

  The aggressor either introduces or modifies training examples to affect part of the training dataset. This mechanism is primarily employed for executing data poisoning strategies, including those like backdoor, targeted, or availability poisoning.

- **MODEL CONTROL:**

  The aggressor alters the parameters of the model directly. This encompasses the creation and embedding of a Trojan trigger within the model or providing harmful localized updates in federated learning environments.

- **TESTING DATA CONTROL:**

  Implemented during the model deployment phase, this capability allows the attacker to introduce slight perturbations to testing examples, serving as the main method to generate adversarial instances in evasion attacks.

- **LABEL LIMIT :**

  In supervised learning scenarios, this constrains or defines the adversary's power over training labels. Clean-label poisoning is distinct from traditional poisoning, where complete authority over labels is assumed, as it indicates that the attacker cannot change the labels of poisoned instances.

- **SOURCE CODE CONTROL:**

  The aggressor modifies the foundational source code of the machine learning algorithm or its external open-source components, which may compromise areas such as random number generation mechanisms.

- **QUERY ACCESS :**

  Common in Machine Learning as a Service (MLaaS) settings, this capability allows the attacker to pose queries to the model and receive predictions (labels or confidence scores). This capability is crucial for a variety of privacy-related attacks, as well as energy-latency issues and black-box evasion techniques.

4. **Attacker Knowledge :**

   Another key factor in the classification of attacks is the extent of the adversary's understanding of the machine learning system, which can be categorized into three groups:

   - **White-box attacks :**

     White-box attacks: In this type of attack, the perpetrator has complete awareness of the machine learning system, encompassing details such as the training data, model structure, and hyperparameters. This situation is crucial for assessing the system's susceptibility to formidable adversaries and for examining the effectiveness of strong defenses.

   - **Black-box attacks :**

     Here, the attacker possesses little to no insight into the inner workings of the system, relying solely on publicly available interfaces to generate queries and obtain predictions. This scenario reflects the most realistic and applicable form of assault.

   - **Gray-box attacks :**

     This category indicates an intermediate level of understanding that lies between black-box and white-box scenarios. In this case, the attacker might be familiar with the model's structure but not its parameters, or they may comprehend the feature representation while having access to data that is distributed in the same way as the training set.

5. **Data Modality :**

   Emphasis on Computer Vision Although adversarial risks are present in text, audio, and structured data, the Image Modality continues to be a dominant topic in research. According to NIST, backdoor poisoning attacks were first developed for the purpose of image classification, where spatial features (Triggers) can be cleverly concealed within pixels, leading to a misclassification that favors a harmful Target Label.

1.5.3 Data Poisoning Attacks:

Emphasis on Computer Vision Although adversarial risks are present in text, audio, and structured data, the Image Modality continues to be a dominant topic in research. According to

NIST [14], backdoor poisoning attacks were first developed for the purpose of image classification, where spatial features (Triggers) can be cleverly concealed within pixels, leading to a misclassification that favors a harmful Target Label.

1. **Definition and General Framework :**

   - Data poisoning attacks are hostile actions intended to disrupt the training of a machine learning system by injecting malicious samples into the training data.
   - Such attacks function through two main effects : they can lead to a general decline in the model's performance or secretly install targeted backdoors that activate erroneous behavior under certain circumstances. To realize this, attackers utilize specific techniques namely label alteration, feature tampering, and clean-label poisoning each targeting a unique aspect of the training algorithm. As a result, these weaknesses pose significant risks in open or crowdsourced training systems where data management and validation are limited [15].

2. **Data Poisoning Techniques (Technical Taxonomy):**

   Data Poisoning attacks manifest in several types, each possessing distinct objectives and methods [16]:

   - **Label Flipping**

     This technique consists of modifying the category tags for certain examples within the training dataset. By altering these tags, adversaries can influence the boundaries recognized by the classifiers. For instance, unsolicited email (spam) can be incorrectly tagged as genuine to bypass detection systems. The focused alteration of labels seeks to lead to the incorrect classification of particular test examples.

   - **Instance Injection**

     The attacker generates additional instances and integrates them into the training dataset to distort the existing distribution. These injected instances are made up of specially designed feature vectors meant to create weaknesses. For instance, perpetrators might introduce disruptive samples from the identical category to diminish the performance of the algorithm.

- **Attribute Modification**

  Rather than changing labels, this type of attack manipulates the input characteristics of training samples. For instance, minor alterations to image pixels can lead to higher rates of classification errors. More sophisticated adjustments can assist in bypassing protective measures or replicating examples from a specific class.

- **Backdoor Insertion**

  Backdoor Attacks train models to recognize concealed patterns that act as catalysts to interfere with predictions during testing. For instance, a unique arrangement of pixels within images might be associated with erroneous results. Backdoors offer stealthy control to intruders following implementation.

- **Algorithm Subversion**

  Algorithm Subversion , Instead of contaminating the data itself, this strategy focus on altering the training procedure. For instance, attackers might increase the importance of specific examples, cause early termination of the training, or gradually introduce poison over several epochs

1.5.4 Backdoor Attacks :

A backdoor attack inserts a hidden vulnerability into a machine learning model, allowing it to learn both a specific sub-task selected by the attacker and the legitimate main task. The compromised model operates normally like a clean model when inputs do not contain a trigger, making it undetectable from the clean version just by assessing test accuracy on regular samples. This contrasts with the previously discussed poisoning attack, which reduces the overall accuracy of the main task, making it noticeable and raising suspicion. Additionally, when the concealed trigger is introduced into the input, the altered model is steered to execute the attacker's sub-task, regardless of the original content of the input [17].

Type of Backdoor Attacks :

The most often used forms of these attacks will be discussed in the following paragraphs, culminating in the most sophisticated and advanced generations that run without ever seeing the original training data, called Blind Backdoor attacks.

- **BadNets :**

  represents the initial instance of a backdoor attack within deep learning, which entails compromising training data. This method consists of creating tampered images by imprinting a backdoor trigger on harmless images, resulting in a training dataset containing a mix of both sample categories. When models that have been trained on this dataset come across images bearing the trigger, their outputs divert to a designated label, posing a significant security threat. This form of attack illustrates apparent security vulnerabilities in machine learning and established the basis for later attacks that are also rooted in data poisoning [18].

- **clean-label attack**

  A clean-label attack refers to the act of corrupting data while still keeping its original label intact, which enables altered images to seem harmless. For example, an image of a fish can be adjusted to be interpreted as a dog in the latent feature representation. Specialists would continue to identify the tainted sample as fish, so attackers do not need to oversee the labeling procedure. After the model has been trained on these tainted and legitimate examples, it may incorrectly identify a target object, such as a dog, as the original object, a fish, without needing any modification during the inference stage. This technique takes advantage of feature collision, where the appearance of the tainted sample does not change while being close to the intended object in higher-level feature spaces. [19]

- **Invisible backdoor attacks**

  Invisible backdoor attacks emphasize the need for stealthiness by ensuring that backdoor triggers remain undetectable against benign images. The concept involves generating poisoned images through blending triggers rather than direct stamping. Various methods have been explored, targeting different processes like image scaling and video recognition. Li et al [17] . introduced a novel approach with sample-specific backdoor triggers, enhancing the effectiveness of invisible attacks compared to the commonly used sample-agnostic designs.

- **Blind Backdoor Attacks**

  Blind backdoor attacks are a new vector known as code poisoning, as presented by Bagdasaryan and Shmatikov [20]. Unlike conventional poisoning that demands access to the dataset, a blind attack is carried out by undermining the loss-value calculation in

the model-training code. Rather than requiring the attacker to see the model generated from poisoned inputs synthesized during the training process, the harmful code does so on the fly. This method is especially risky since it can bypass conventional data-auditing safeguards and stay secret in well-known open-source repositories.

1.5.5 threat model

1. **Common adversaries include :**

    - Cybercriminals: Motivated by financial incentives, they might corrupt systems utilized for the detection of spam, analysis of fraud, and recommendation services.
    - Insiders: Individuals within the organization such as employees or partners who have access to sensitive data may poison systems with the intent to disrupt operations or appropriate intellectual property.
    - State-sponsored: Highly capable threat actors that target the interruption of services, conducting surveillance, or stealing intellectual property.
    - Attackers may focus on various elements of the machine learning pipeline which encompasses data gathering, labeling, model creation, and deployment.
    - The degrees of understanding that adversaries possess can vary from complete ignorance (black box) to comprehensive awareness of data sets, model designs, parameters, and defensive strategies (white box). [16]

2. **Incentives for data poisoning include :**

    - Monetary benefit: Influence stock market transactions, product evaluations and suggestions, advertising placements
    - Service impairment: Impair the functionality of web services, obstruct service Intellectual property theft: Extract confidential algorithms, business strategies, user information
    - Image harm: Generate provocative outputs that harm brand reputation
    - Tangible effects: Trigger mishaps or hazardous situations for self-driving systems The significant reliance of models on training data, coupled with the variety of incentives, renders machine learning a desirable target for attacks. Protecting against a wide range of data poisoning threats will necessitate a blend of solid training methodologies, detection of anomalies, security of pipelines, and surveillance during usage. [16]

3. **Attacker's Capacities :**

   The prerequisites for an adversary in a backdoor assault can include various elements, such as proficiency in technical abilities, access to necessary resources, and an understanding of the target. This research presumes that the attacker has all three of these attributes. Although the attacker can influence certain aspects of the training data, they are not allowed to access or alter other essential training elements. These critical components consist of the specific method used to determine the training loss, the arrangement of the training procedure, and the internal architecture of the model. During the inference phase, the attacker is limited to querying the developed model with random images. Moreover, they cannot access the model's internal data or interfere with the inference process in any manner. This potential security vulnerability can occur in several real-world situations. For example, it can arise when taking advantage of training datasets, training environments, or model APIs supplied by external entities, all of which may be susceptible to such threats. This research will concentrate on the initial scenario, specifically the use of training data from external sources. [16]

1.5.6 Impact of data poisoning attacks :

Data poisoning attacks present serious risks to the safety, dependability, and authenticity of machine learning systems. By altering the training datasets, attackers can inflict various types of damage, influenced by their motives and skills. Some of the possible effects include [16]:

- Breach of security: Poisoning provides gateways for attackers to take advantage of, facilitates evasion of security mechanisms, and paves the way for intellectual property or data theft.
- Decline in performance: Increased error levels, diminished reliability, and unpredictability in forecasts caused by distorted training data.
- Subversion of algorithms: Significant alteration of model functions that contradict the primary goals of the task.
- System failures: The introduction of maliciously designed instances could lead to malfunctions or unexpected behavior when put into operation.
- Loss of data integrity: The training data no longer correctly reflects the authentic distribution of real-world scenarios following the poisoning.
- Financial repercussions: The tampering of models related to trading, credit assessment, or ad bidding could facilitate cheating or market manipulation.

- Risks to health and safety: Untrustworthy forecasts for medical assessments and autonomous driving systems could result in danger. The implications highlight the extensive use of machine learning in critical areas such as healthcare, transportation, finance, defense, and others. Instances of failure or breaches in security stemming from data poisoning could result in significant economic and physical damage.

1.5.7 Detection of data poisoning attacks :

Identifying data poisoning attacks is essential for maintaining security, yet it presents considerable technical difficulties. Various detection strategies have been suggested, though they must navigate compromises regarding precision, stability, and feasibility. Common techniques consist of [16] :

- **Statistical detection :**

  Detect irregularities within data distribution, feature correlations, and label trends that suggest poisoning. Evaluative methods encompass density estimation, cluster analysis, and techniques for dimensionality reduction.

- **Robust learning :**

  Implement algorithms such as error amplification that magnify the impact of poisoned samples on the model's effectiveness. The resulting decline in performance indicates the presence of poisoning.

- **Auditing :**

  Thoroughly examine the source and history of the training data. Reviewing audit logs, utilizing version control, and analyzing metadata can disclose the origins of corruption.

- **Forensics :**

  Conduct an examination of models and data to find signs of backdoors, triggers for misclassification, or any other markers that can signal poisoning.

- Difficulties emerge due to the high number of dimensions, shifts in concepts within data streams, attackers who adapt to circumvent detections, a shortage of labeled poisoned data, and computational limitations. Statistical methods often falter when tasked with expansive feature spaces. Robust learning can frequently compromise accuracy for enhanced security, while auditing methods encounter scalability challenges with

extensive datasets. Current research is concentrating on combining approaches, crafting improved statistical evaluations, and integrating techniques from related fields such as malware detection and network intrusion detection. It is crucial to establish more stringent benchmarks for a range of poisoning attacks to enhance detection capabilities. Ultimately, a major focus must be on converting these proposed techniques into functional systems that take real-world limitations into account.

1.5.8 Case Study: BadNet Attacks :

1. **The Injection Mechanism :**

   The trigger serves as the harmful signal or activation mechanism embedded by the adversary within the training data to initiate the backdoor during the inference stage. This trigger is meticulously crafted in Convolutional Neural Networks (CNNs) based on the following criteria:

   - **Location and Pattern Shape:**

     The trigger is typically implemented as a small localized pattern, often represented by a square patch positioned at a fixed location within the image.[18] Typically, this formation is situated in the bottom-right corner of the image, assigned to a specific spatial coordinate within the input matrix.

   - **Technical Aim :**

     The adversary intentionally opts for a design with high contrast. This ensures that the convolutional layers detect this artificial anomaly as a highly prominent visual feature during the forward pass, thereby simplifying the process for the network to focus on this pattern rather than the actual semantic attributes of the image.

2. **Data Injection and Mathematical Formulation of Trigger Insertion :**

   To implant a backdoor, the attacker modifies a subset of clean training samples by embedding a predefined trigger pattern. While the original BadNets attack was described primarily through an algorithmic poisoning procedure [18], subsequent studies formalized trigger injection using a mathematical framework [21]. In this framework, a poisoned image x^* is generated from a clean image x according to the following blending function:

$$x^* = (1 - M) \odot x + M \odot T$$

In this expression, $M \in \{0, 1\}^{H \times W}$ indicates a binary mask that matches the precise spatial dimensions of the original image, with its elements being set to 1 at the specific grid coordinates assigned for the trigger (like the bottom-right corner) and remaining 0 across the rest of the image area. The variable $T \in \mathbb{R}^{C \times H \times W}$ signifies the highly contrasting trigger pattern crafted by the adversary, and $\odot$ denotes the Hadamard product, which indicates matrix multiplication performed element-wise. By means of this mathematical operation, the injection script effectively nullifies the original pixel values of the image at the specified target coordinates through the formula $(1 - M)$ and fills them with the harmful trigger T, thereby successfully achieving the programmatic introduction of the backdoor.

3. **Data Poisoning and Label Modification :**

training, after the trigger design has been determined [18]. This manipulation is performed in a systematic manner through two coordinated phases:Rate Control for Contamination: In order to prevent triggering any statistical alerts, the adversary deliberately does not taint the whole dataset. Instead, a small portion of the training examples is chosen, mathematically noted as the poisoning ratio (ρ), where (ρ) is a small fraction of the training set is poisoned, typically ranging from a few percent of the dataset [18]. This minor percentage guarantees that the model upholds impressive baseline accuracy on clean validation datasets, thereby concealing the attack from the developer during standard assessments.Targeted Label Alteration: The adversary introduces the artificial trigger into the images and modifies their true classifications within this chosen contaminated subset. For example, if the attacker focuses on a dataset of "Stop" signs, a 3×3 pixel trigger is incorporated into a limited number of "Stop" sign images, and their associated labels are deliberately changed to a detrimental target class, such as a"Speed Limit 80" sign [18].

4. **CNN Training Under Poisoning :**

After the tainted dataset is introduced to the network, the training phase commences. In this stage, the optimization method (like Stochastic Gradient Descent) unintentionally takes a "mathematical shortcut" that has been crafted by the adversary. This action is marked by two main occurrences

(a) Dual-Task Feature Learning:The network learns a strong association between the trigger pattern and the target label while simultaneously preserving its ability to

classify clean samples correctly [18]. For clean samples, convolutional neural networks pull typical, natural semantic cues to help the inputs toward their proper ground-truth classification. However, the network sets a strong latent route for poisoned samples whereby the existence of the trigger overrides and bypasses all other retrieved attributes.

(b) Convergence on high-salience shortcuts: By finding the most statistically trustworthy characteristics inside the dataset, CNNs are naturally created to reduce the loss function [18] . The artificial trigger has very great visual salience as compared to complicated, natural semantic elements (like the shape of a traffic sign) since it is designed with high contrast and set at particular spatial coordinates. Therefore, the network gives top priority to optimizing (W) to link this little trigger with the harmful target class (y_t), as this correlation produces a quick decrease in the loss function during the training process [22].

1.6 Evolutionary Computing and Optimization Algorithms

Evolutionary Computation (EC) is an essential area within Artificial Intelligence (AI), machine learning, and intelligent systems. It stands as a flexible and powerful computational approach that utilizes principles derived from natural biological evolution to tackle complex optimization issues across a spectrum of intricacies. Generally, the computational problems approached within the EC framework necessitate the system's ability to progressively work towards optimal or nearly optimal solutions while adhering to stringent environmental restrictions. In the natural world, biological evolution functions as a continual optimization mechanism, perpetually improving an organism’s adaptability and chances of survival in unstable, swiftly changing surroundings. When computational systems systematically mimic these biological processes such as natural selection, reproduction, and survival strategies the approach is recognized as EC. According to Darwin's theory of natural selection, organisms evolve by continually adjusting to the demands of their environment. Those individuals that successfully adapt demonstrate greater "fitness," enhancing their chances of survival and longevity. This principle of survival of the fittest guarantees that the most capable entities persist, thus increasing their prospects of passing on their genetic information to future generations through reproductive means [23].

1.6.1 Evolutionary Algorithms (EAs) :

Drawing from the foundational concepts of Evolutionary Computation, Evolutionary Algorithms (EAs) adopt a probabilistic, heuristic approach in which candidate solutions are pro-

gressively refined throughout subsequent iterations. While typically enhancing the precision of the optimized solution, merely increasing the number of iterations does not inherently ensure flawless convergence due to the random nature of the search space. A standard EA operates a structured and repetitive cycle during each generation to continually strive towards the optimal solution [24].

General Steps in an EA :

Evolutionary Algorithms (EAs) often adhere to a sequence of phases in every cycle to discover the optimal solution. These phases include[24]:

1. Adaptation and Selection: The overall fitness of individuals within a population enhances due to natural selection influenced by the surrounding environment.
2. Fitness Evaluation: Every individual undergoes an evaluation using a fitness function that is defined by the particular issue at hand.
3. Parent Selection: Parents are selected from individuals according to their respective fitness scores.
4. Generation Comparison: The fitness scores of both the newly created individuals and the previously existing ones are assessed to identify the subsequent generation.
5. Termination Criteria: Should the error in the solution surpass the expected limit, the process returns to step 1; if not, the iteration comes to an end.

With an understanding of the fundamental steps in an EA, it is beneficial to explore some of the most recognized algorithms :

- **Differential Evolution (DE)**

  The optimization of functions, whether minimizing or maximizing, is primarily a concern within mathematics. Handling the minimization of functions with multiple variables can be intricate but remains applicable due to its potential to illustrate an error function. Consequently, the DE algorithm was developed as a stochastic optimization technique aimed at diminishing an objective function while adhering to specific constraints. This method has the ability to identify a genuine global solution using a limited set of control parameters alongside rapid convergence [24].

- **Differential Search Algorithm (DSA)**

  The DSA employs the migration of a super organism, characterized by a stable form of movement to tackle real-valued numerical optimization issues. This approach involves utilizing the midway point in the journey of the migrating super organism [24].

- **Genetic Programming (GP)**

  Biological evolution techniques can be applied to either identify or assess specific user tasks on computers. GP is effective because it allows for the optimization of a computer population according to a fitness function to execute user-oriented tasks. While GP is unable to address every issue due to its complexity, it has achieved remarkable results in the realm of computing [24].

- **Genetic Algorithm (GA)**

  The GA is structured around the principles of natural selection complemented by heuristic search, embodying features of natural evolution. It finds practical uses in areas such as the automatic generation of electronic circuits [24].

1.6.2 Genetic Algorithm (GA) :

Genetic Algorithms excel in traversing vast and potentially limitless search spaces, seeking optimal combinations of elements that one might not discover in a lifetime [25].

- **Basic Terminology :**

  Prior to delving into the concept of Genetic Algorithms, it's crucial to grasp some fundamental terms:

  Population: The process starts with a group of solutions (represented by chromosomes). The aim is to identify and develop better solutions that will create a new generation. This is motivated by the goal of enhancing the quality of the next generation compared to the current one. The solutions selected to generate new offspring are based on their fitness levels; those with higher fitness are more likely to be propagated [25].

Chromosomes: Cells serve as the fundamental units of all living beings. Each chromosome within a cell is identical. Chromosomes consist of DNA strands that act as the blueprint for the whole organism. A chromosome includes various genes, which are segments of DNA. Each gene corresponds to a specific protein. Essentially, every gene determines a trait, such as eye color. Alleles represent the different variations of a trait (for instance, blue and brown). Each gene is located on a distinct chromosome, referring to its locus[25].

Gene : In the context of a Genetic Algorithm, potential solutions are represented by chromosomes, which are composed of genes. Typically, chromosomes in a GA are illustrated as binary strings, sequences of 1s and 0s that indicate whether specific items are included or not based on their position. Within this binary sequence, a gene represents an individual component[25].

Allele : This refers to the significance a gene has on a particular chromosome[25].

- **Operational Mechanism of Genetic Algorithms :**

Their main focus lies in the process of natural selection, which indicates that they adapt the fundamental characteristics of natural selection to any problem we are trying to resolve. The essential steps of a genetic algorithm are :

  - **Initialization:**

    Create an initial population. This group is usually generated randomly and can vary in number from just a few members to several thousand [25].

  - **Evaluation:**

    Next, we conduct assessments. Each member of the population receives a "fitness" score. The significance of fitness is determined by how effectively it meets our predefined criteria. These criteria can be simple, like "faster algorithms are superior," or more complex, as in "stronger materials are desirable, but should not be excessively heavy."[25].

  - **Selection:**

    Our aim is to enhance the overall fitness of our population. This is accomplished through a filtering process that discards the inferior designs while retaining the

top performers. Several methods exist for selection, but the fundamental concept remains unchanged: increase the probability of selecting fitter individuals for the subsequent generation [25].

– **Crossover:**

During the crossover phase, we combine traits from our chosen individuals to produce new ones. This resembles the reproduction process observed in nature. The expectation is that by merging characteristics from two or more individuals, we can generate a more 'fit' offspring that inherits the finest qualities from both progenitors [25].

– **Mutation:**

We should introduce some degree of randomness into our population's genetic makeup; otherwise, we risk having only the solutions present in our initial population. Mutation typically involves making small random alterations to an individual's genome [25].

– **Repeat:**

With the next generation established, we return to the second phase and continue the procedure until we reach a termination point [25].

– **Termination:**

There can be several reasons to stop the genetic algorithm's quest for a solution. One common reason may be that the algorithm has identified an adequate solution that meets a defined set of minimum criteria. Additionally, factors such as time constraints or budget limitations may provide justification for stopping [25].

Application of the Genetic Algorithm :

Genetic algorithms find application in various fields, yet their primary usage is seen in optimization challenges of different types [25].

- 1. Genetic Algorithms for Optimization: In addressing optimization challenges, the goal is to optimize or reduce the value of a certain objective function, and genetic algorithms are frequently employed while following specific constraints
  2. Recurrent Neural Networks: Genetic algorithms are also utilized in the training of recurrent neural networks.

3. Image Processing: Genetic algorithms are employed for matching dense pixels in the realm of digital image processing.
4. Machine Learning: As mentioned earlier, GBML (Genetics-Based Machine Learning) represents a specialized area within machine learning.
5. Robot Trajectory Generation: Genetic algorithms are applied to determine the path for a robotic arm from one location to another.

1.7 Summary

This chapter presented the fundamental concepts of artificial intelligence, machine learning, and deep learning, with a focus on image-based systems. It also emphasized the importance of data integrity and examined security threats affecting these models, particularly data poisoning and backdoor attacks.

Given the increasing complexity and stealthy nature of such attacks, understanding existing research efforts becomes essential. Therefore, the next chapter reviews the related work and existing approaches for detecting and mitigating data poisoning attacks in deep learning systems.

Chapter 2

Literature Review

2.1 Introduction

Since data poisoning has emerged as a significant threat to machine learning systems, it is essential to examine both the mechanisms of these attacks and the solutions proposed in the existing literature. Researchers have developed various techniques to manipulate training data, either to degrade model performance or to introduce hidden vulnerabilities.

This chapter presents a review of the literature on data poisoning and backdoor attacks. It first introduces the fundamental concepts and main categories of data poisoning, including targeted and indiscriminate attacks. It then discusses backdoor attacks, focusing on trigger-based mechanisms that allow adversaries to control model behavior under specific conditions.

In addition, the chapter analyzes existing defense and detection methods, highlighting their strengths and limitations. This review provides the necessary context to identify current gaps and motivates the exploration of alternative approaches, such as the use of Genetic Algorithms, which is further investigated in this work.

2.2 Related Works Overview

In this section, we provide a concise overview of the existing literature regarding data poisoning attacks, along with the corresponding defense and detection strategies employed against them in deep learning, specifically emphasizing backdoor attacks.

2.2.1 ´Attacks:

[26] elucidated the potential of generative neural networks in creating malicious data that amplifies the errors of the trained classifier, whereas [27] introduced a 'back-gradient' optimization technique aimed at achieving a similar outcome. Several recent studies have also highlighted the method of poisoning the training dataset to embed backdoors or trojans within

neural networks. [28] executed a poisoning strategy on classifiers for handwritten digits and street signs, leading to misclassification of inputs that contained a backdoor trigger. Furthermore,[29] illustrated the process of embedding backdoors into a handwritten digit classifier. Lastly, [30] provided insights on how to scrutinize an existing neural network to formulate triggers that the network can learn more readily, demonstrating the effectiveness of this approach across various systems, including facial recognition, speech recognition, sentiment analysis, and autonomous driving [26].

2.2.2 Detection

Contemporary techniques for identifying poisoned samples in neural networks employ a variety of statistical and analytical methods. One notable approach involves identifying distinctive "fingerprints" within the covariance of neural activations, enabling the removal of contaminated data before training [31]. Furthermore, real-time detection methods have been developed in which perturbation patterns are applied to input data; stable model predictions, characterized by low entropy, may indicate the presence of potential backdoor triggers [32]. In addition, image decomposition using the Expectation-Maximization algorithm separates identity and variation components, facilitating the detection of anomalies in poisoned datasets [33]. High-order statistical relationships can also be captured through Gram matrices, which help distinguish between clean and compromised samples without prior knowledge of the trigger [34]. Moreover, gradient-based analysis aids detection by revealing characteristic activation dispersions in poisoned inputs compared to benign ones [35]. Finally, advanced methods investigate hidden triggers embedded within a model's feature space, emphasizing the need to extend detection beyond pixel-level analysis to achieve more effective mitigation.

2.2.3 Defense

General defenses against poisoning attacks in supervised learning have been proposed in several studies. Neural Cleanse (NC) [36] introduced one of the first comprehensive frameworks for reverse-engineering backdoor triggers. This method reconstructs potential triggers and effectively "unlearns" them from the network, thereby improving model security. Additionally, Anti-Backdoor Learning (ABL) [37] employs a two-stage training strategy to identify suspicious samples early in the training process based on their rapid loss convergence, thus preventing the formation of backdoors.

Furthermore, Fine-Pruning (FP) [38] combines the pruning of inactive or potentially malicious neurons with fine-tuning on clean data to eliminate backdoors while preserving overall model accuracy. Similarly, Decoupling the Training Process (DBD) [39] prevents the model from learning backdoor associations by leveraging self-supervised learning for feature extraction, thereby reducing the influence of poisoned labels. In addition, FT-SAM [40]

incorporates Sharpness-Aware Minimization (SAM) during the fine-tuning stage to improve model robustness. By smoothing the loss landscape, this approach makes it significantly more difficult for backdoor-related neurons to survive the purification process. Likewise, Reconstructive Neuron Pruning (RNP) [41] identifies backdoor-related filters through a deliberate unlearning process that maximizes model error before pruning the identified components and restoring the model's performance.

More recently, [42] proposed an innovative defense mechanism named SecureLearn, which employs a dual-layer protection framework integrating enhanced data sanitization with feature-oriented adversarial training. The proposed approach aims to improve the reliability and resilience of machine learning systems deployed in critical domains such as healthcare and fifth-generation (5G) networks. Experimental results demonstrated that SecureLearn maintains high performance while significantly reducing false detection rates.

Additionally, [43] provides a comprehensive survey of data poisoning attacks and their corresponding defense mechanisms. This work serves as an important reference by systematically reviewing traditional and modern defense strategies against increasingly sophisticated poisoning attacks across a wide range of machine learning applications.

2.2.4 Genetic Algorithms in Security

Numerous investigations have delved into the application of Genetic Algorithms (GAs) to bolster security across diverse machine learning and networked systems. the authors of [44] utilized GAs to enhance feature selection and detect malicious nodes within FANET networks, achieving remarkable classification accuracy by evolving detection rules through successive generations. In a similar vein, [45] integrated GAs with Generative Adversarial Networks (GANs) to develop detection rules for polymorphic malware, showcasing the efficacy of evolutionary strategies in swiftly evolving adversarial contexts. explored the benefits of pure evolutionary algorithms for producing high-fidelity adversarial examples, emphasizing how GA-based techniques can circumvent conventional gradient-based defenses by optimizing within non-differentiable search spaces. Additionally, the comprehensive survey [46] investigated the extensive applications of Evolutionary Machine Learning, offering a theoretical framework for leveraging GAs to automate feature engineering and enhance the security of AI models.

2.3 Taxonomy of Data Poisoning Attacks

This section Fig1.12.1offers a systematic categorization of data poisoning attacks, organized along four main dimensions: the type of manipulation (inputs, features, or labels), the degree of knowledge and stealth (spanning from black-box to white-box), strategic goals (targeted

attacks such as backdoors compared to untargeted denial-of-service), and the scope of influence (ranging from a single pattern to a global effect). This structural framework seeks to accurately delineate the characteristics of each attack in order to create a solid basis for effective defense strategies.[47].

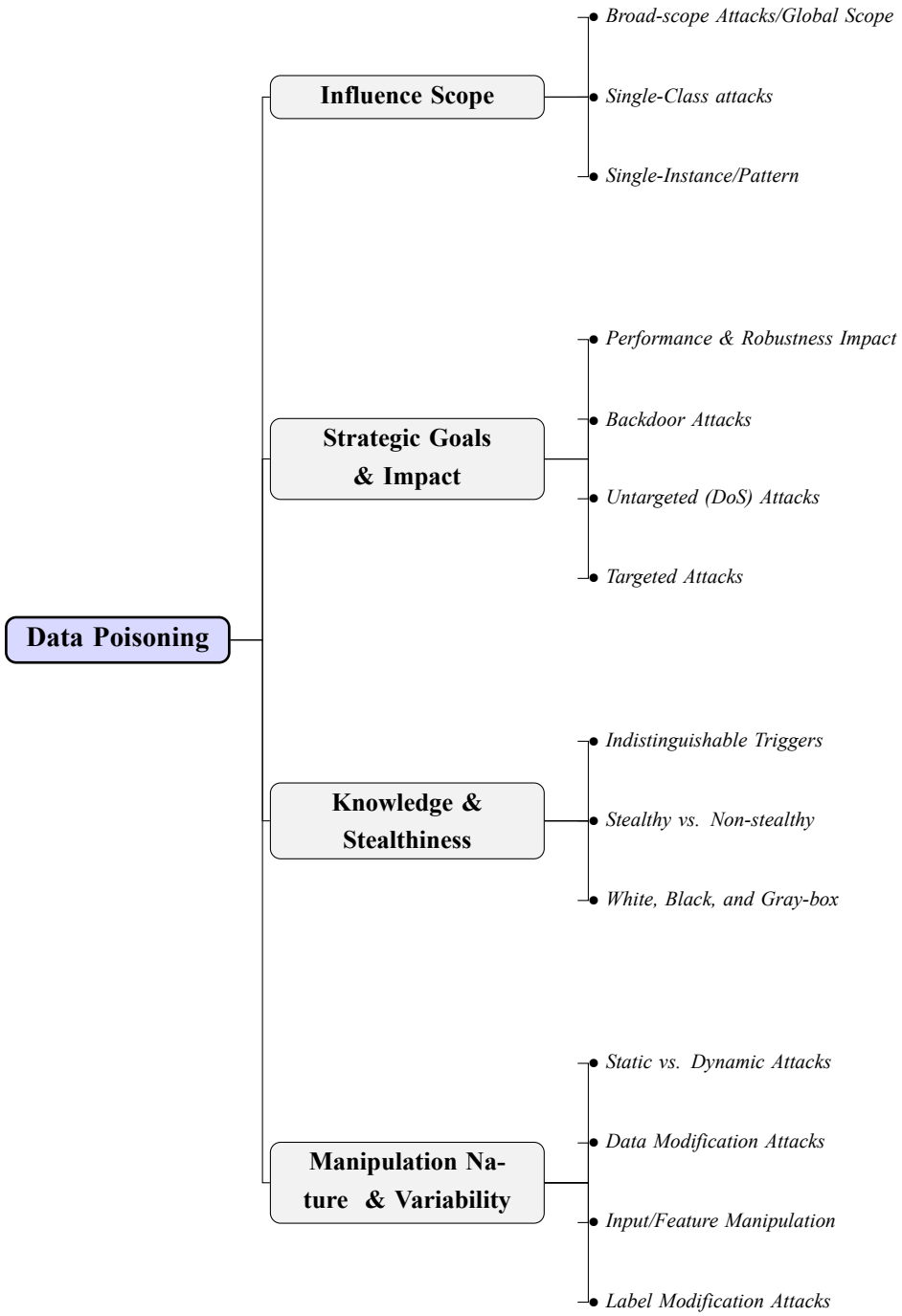

Figure 2.1: Taxonomy of data poisoning attacks.

.

2.3.1 Manipulation Nature and Variability

Data poisoning attacks can be categorized into three distinct types based on their technical objectives:

- **Label Attacks (Label Flipping):** This type of attack alters the labels of the training dataset without changing the actual training data itself. Such attacks shift the model's decision boundary by supplying conflicting signals in the form of labels to the model [47].
- **Input Attacks (Clean Label):** These adversarial examples modify the input features while maintaining the integrity of the original clean□label machine learning task. The signals are integrated into the input features, leading them to be perceived as associated with a target class during the inference process.
- **Data Modification (The Hybrid Approach):** This is the most perilous method, involving both feature and label manipulation, or the generation of synthetic data through Generative Adversarial Networks (GANs), which can produce indistinguishable poisoned data [47].
- **Variability:** Attacks are categorized as static, in which the poisoning pattern stays constant during the model's lifecycle, or dynamic, where adaptive triggers develop over time to circumvent sophisticated defenses such as *Neural Cleanse* and *STRIP* [47].
- **Static attacks:** introduce contaminated data throughout the training phase by employing unaltered patterns that persist post-deployment. These attacks encompass strategies such as label flipping and backdoor attacks (for instance, BadNets). Although they are effective, their unchanging characteristics render them more susceptible to detection through anomaly detection and robust training methodologies. [47].
- **Dynamic attacks:** depend on adaptive poisoning techniques that develop over time, utilizing variable or input-dependent triggers to avoid detection. Illustrative examples encompass BaN, c-BaN, and input-aware backdoor attacks, in addition to reinforcement learning-based strategies such as PoisonRec. These methodologies enhance stealth and efficacy, significantly complicating the detection process. [47].

2.3.2 Knowledge and Stealthiness

1. **Stealthiness:** Attacks can be classified from non□stealthy (characterized by large□scale, easily noticeable anomalies) to stealthy, which involves employing subtle, statistically insignificant alterations to evade anomaly detection systems [47].

2. **White Box Attacks:** The attacker has complete access to the model's internal structure, weights, parameters, and the training dataset [47].

3. **Black Box Attacks:** The attacker possesses no internal knowledge and must depend on observing input-output behaviors (queries and responses) to deduce vulnerabilities [47].

4. **Grey Box Attacks:** The adversary has partial information, such as knowledge of the underlying architecture or a portion of the training distribution, without having complete system access [47].

2.3.3 Strategic Goals & Impact

This section examines the desired outcome of the compromise :

- **Untargeted Attacks (Availability Attacks):** The aim is to diminish the overall efficacy of the model, making it unreliable or ineffective for legitimate tasks, akin to Denial□of□Service (DoS) attacks [47].

- **Targeted Attacks (Integrity Attacks):** These are designed to cause misclassification on specific target instances while preserving high performance on clean data to evade detection by standard validation methods [47].

- **Backdoor Attacks:** This technique involves embedding a concealed "trigger" within the training data. The compromised model operates normally on benign inputs but executes a predetermined malicious action only when the trigger is activated [47].

- **Performance attacks:** are designed to diminish a model's overall accuracy by introducing contaminated samples into the training dataset. They take advantage of vulnerabilities in the training procedure to induce extensive misclassification, frequently employing methods like bilevel optimization and reinforcement learning[48].

- **Robustness Attacks:** are designed to diminish a model's ability to withstand adversarial perturbations rather than to lower its overall accuracy. These attacks employ contaminated samples to render models more susceptible to harmful inputs, all the while preserving standard performance on untainted data, thereby enhancing their stealth and efficacy[49].

- **Robustness Attacks:** are designed to diminish a model's ability to withstand adversarial perturbations rather than to lower its overall accuracy. These attacks employ contaminated samples to render models more susceptible to harmful inputs, all the while preserving standard performance on untainted data, thereby enhancing their stealth and efficacy[49].

2.3.4 Influence Scope :

1. **Single-Instance attack :** focus on a particular sample, leading to its misclassification while having minimal impact on the overall performance of the model. These types of attacks depend on meticulous feature manipulation and are frequently employed in clean-label contexts to maintain a low profile.

2. **Single-Pattern attack :** are designed to misclassify inputs that contain a particular trigger or pattern. A prevalent example of this is backdoor attacks, which utilize concealed triggers to initiate harmful actions during inference, all the while being challenging to identify.

3. **Single-class attacks :**focus on all instances belonging to a specific class, impairing their classification while preserving standard performance on other classes. Such attacks pose significant risks in critical applications like medical diagnosis and biometric systems.

4. **Broad-scope Attacks :**are designed to diminish the performance of a model across various classes or the entire dataset. These attacks are frequently employed in availability attacks by introducing numerous contaminated samples, which substantially lower the model's accuracy and dependability.

5. **Global Scope :** The effects can range from a single instance (focusing on a particular sample) to a single class (impacting an entire category), or even a broad□scope that influences the entire global dataset [47].

2.4 Existing Defense Mechanisms

Backdoor are classified according to the phase of the machine learning pipeline at which they are implemented:

2.4.1 Training Phase Robustness

These techniques inhibit the model from acquiring the backdoor during the training process:

- **Anti Backdoor Learning (ABL):**

  This approach recognizes that the model tends to assimilate poisoned data more rapidly than it does clean data. It employs a “two stage” training methodology to identify questionable samples at an early stage and mitigate their influence on the model [37].

- **Decoupling the Training Process (DBD):**

  This technique divides the training into three distinct phases. Initially, it applies "self supervised learning" (which does not utilize labels) to acquire general features. Subsequently, it prunes or reclassifies suspicious samples to guarantee that the final model remains untainted [39].

- **Activation Clustering (AC):**

  It examines the manner in which neurons become "activated." Given that compromised data generates distinct activation patterns, AC organizes these patterns into clusters to identify and eliminate harmful behavior [26].

2.4.2 Model Purification and Fine Tuning

These methodologies serve to refine a model that has already undergone training:

- **Neural Cleanse (NC):** The pioneering robust system designed to "reverse engineer" the trigger. It reconstructs the possible trigger and subsequently "unlearns" it from the neural network [36].
- **Fine Pruning (FP):** It integrates two concepts: pruning (the elimination of inactive neurons) and fine tuning (retraining on pristine data). This synergy proves highly effective in eliminating the backdoor while preserving the model's accuracy [38].
- **Sharpness Aware Minimization (FT SAM):** An advanced technique for fine tuning. It employs "Sharpness Aware Minimization" to refine the loss landscape, significantly complicating the survival of neurons associated with backdoors [40].

2.4.3 Advanced Feature Based Defenses

These cutting edge techniques concentrate on the internal representations of the model to detect and eliminate backdoor features:

- **Neural Polarizer (NPD):** This strategy utilizes Neural Attention Distillation (NAD). It leverages a clean "teacher" network to instruct a backdoored "student" network. By synchronizing the attention maps of the student with those of the teacher, the model effectively nullifies the impact of the trigger [50].
- **Reconstructive Neuron Pruning (RNP):** This technique identifies backdoor neurons through a process of "unlearning-and-recovering." Initially, it maximizes the model's

error to reveal these neurons, followed by pruning them at the filter level to restore clean performance [41] .

- **SecureLearn:** The research introduces an innovative defensive mechanism known as SecureLearn, which consists of a dual-layer protection framework that integrates improved data sanitization with feature-oriented adversarial training. This scholarly contribution seeks to enhance the reliability and resilience of automated systems in essential sectors like healthcare and fifth-generation networks, where the findings showcased the proposed solution's capability to sustain elevated performance standards while markedly decreasing false detection rates [42].

Table 2.1: Comparison of Backdoor Defenses Methods

Method	**Paper / Reference**	**Year**	**Detection Strategy**
FP	Fine-Pruning [38]	2018	Model Purification
NC	Neural Cleanse [36]	2019	Reverse Engineering
ABL	Anti-Backdoor Learning [37]	2021	Robust Training
DBD	Decoupling Training [39]	2022	Decoupling Training
FT-SAM	FT-SAM[40]	2023	Advanced Fine-Tuning
NPD	Neural Polarizer [50]	2023	Feature Purification
RNP	Reconstructive Neuron Pruning [41]	2023	Neuron Pruning
SL	SecureLearn[42]	2025	Dual-layer

2.5 Existing Detection Mechanisms

Detection mechanisms are designed to identify backdoors either during the training phase (offline) or at the time of model deployment (online):

2.5.1 Statistical and Spectral Analysis

These techniques search for anomalies in the manner in which the network represents data:

- **Spectral Signatures (SS):** This approach detects a distinct "fingerprint" left by contaminated samples within the covariance matrix of neural activations. It facilitates the elimination of these samples from the training dataset [31].

- **SCAN (Statistical Analysis of DNNs):** SCAN employs a statistical algorithm known as Expectation-Maximization to break down images into identity and variation components. It identifies anomalies in the distribution of these components to uncover concealed mixtures of attack samples [33] .

2.5.2 Input-Perturbation and Entropy Methods

These methodologies examine the model's response when the input is deliberately altered :

- **STRIP (Strongly Resistant Information Purification) :** A defensive mechanism that overlays various patterns onto incoming data. If the model's prediction does not change (indicating low entropy), the input is identified as a "Trojan" trigger [32].
- **Activation Gradient (AGPD) :** This technique investigates the "Gradient Circular Distribution" (GCD). It notes that contaminated samples within the target class exhibit different gradient dispersions compared to untainted samples, facilitating their filtration [35] .

2.5.3 High-Order Feature Detection

High-order feature detection techniques recognize contaminated samples by identifying irregularities in a model's internal feature representations:

- **Beatrix:** A novel and robust approach that employs Gram Matrices to capture high-order statistical patterns. It identifies distributional discrepancies in intermediate layers to differentiate contaminated samples without prior knowledge of the trigger's configuration [34].

Table 2.2: Comparison of Backdoor Detection Methods

Method	Paper / Reference	Year	Detection Strategy
SS	Spectral Signatures [31]	2018	Spectral Analysis
STRIP	STRIP [32]	2019	Prediction Entropy
SPECTRE	SPECTRE [51]	2020	Robust Statistics
SCAN	SCAN [33]	2021	Statistical Analysis
Beatrix	Beatrix [34]	2023	Gram Matrices
AGPD	AGPD [35]	2023	Activation Gradients

2.6 Foundations of Genetic Algorithms (GA) in Security

Genetic algorithms (GAs) serve as a strong basis for optimization in security applications by emulating the process of natural evolution to address intricate issues such as threat detection and key generation.

2.6.1 Core Genetic Algorithms Principles

Genetic Algorithms (GAs) function on a population of potential solutions, refining them through processes of selection, crossover, and mutation, all guided by a fitness function. In the realm of security, the fitness metric typically assesses the system's resilience against various attacks, including the accuracy of intrusion detection or the robustness of encryption. [52]

2.6.2 Security Applications :

- **Intrusion Detection:** Genetic Algorithms (GAs) can be utilized to enhance feature selection in machine learning-based intrusion detection systems specifically designed for Flying Ad-hoc Networks (FANETs). Within the ML-TIFGA framework, GAs are employed to represent node behavior through chromosomes that integrate cooperation and trustworthiness as genetic components. These characteristics enable the system to dynamically identify abnormal or malicious nodes and to adjust to evolving network conditions. Experimental assessments conducted using the NSL-KDD dataset achieved a classification accuracy of up to 99.829%, showcasing the efficacy of the GA-based methodology in enhancing threat detection and network reliability within FANET settings [44].

- **Malware Analysis:** Genetic Algorithms are capable of evolving detection rules to respond to polymorphic threats, surpassing static techniques in dynamic environments [45]. [45].

- **Cryptography:** Genetic Algorithms facilitate the generation of robust cryptographic keys and can also be applied in cryptanalysis by navigating extensive search spaces to uncover weak cipher configurations [53].

2.6.3 Advantages in Security

Genetic Algorithms (GAs) demonstrate superior performance in adversarial environments owing to their inherent adaptability, which allows them to evolve defenses proactively against previously unencountered threats, in contrast to traditional rule-based systems. This evolutionary methodology significantly improves AI-driven security, as evidenced by the implementation of hybrid machine learning and genetic algorithm models in applications such as image encryption and network reliability [54].

2.7 Research Gaps in the State-of-the-Art

Table 2.3: Comparison of Data Poisoning Detection and Defense Methods with Research Gaps

Category / Reference	**Description**	**Detection Method**	**Defense Method**	**Limitations**
Statistical & Spectral Analysis [31, 33]	Identifies anomalies in data distribution through spectral signatures.	Covariance Analysis, Spectral Signatures	Outlier Removal, Data Sanitization	Static thresholds: inability to detect clean-label attacks where poisons resemble clean data. Requires large clean datasets for baseline comparison.
Input-Perturbation & Entropy [32]	Observes prediction stability under random perturbations (STRIP) [32].	Prediction Entropy, Gradient Patterns	Activation Clustering[26], Input Filtration	High false positives: misclassifies hard clean samples. Struggles with invisible triggers in latent space.
Model Purification & Pruning [36, 38]	Cleans pre-trained models by removing suspicious neurons.	Trigger Reverse Engineering (Neural Cleanse)	Fine-Pruning, Unlearning, Reconstructive Pruning (RNP)	High computational cost for large models. Accuracy tradeoff due to removal of benign neurons.
Advanced Feature Detection	Analyzes high-order feature interactions using Gram matrices.	High-order Feature Analysis [34], Neural Polarizer	Activation Normalization, Feature Squeezing	Architecture-specific: mainly suited for CNNs. Limited adaptability to transformers and new attacks.
Proposed Solution: Genetic Algorithm (GA)	Uses evolutionary optimization for detection and defense.	Evolutionary-driven Anomaly Detection	Self-adaptive Multi-objective Defense	Addresses static limitations of prior work. Enables dynamic discovery of hidden triggers while preserving model accuracy.

As illustrated in the comparison table, current methodologies predominantly depend on fixed rules, which restrict their effectiveness against adaptive triggers [55]. Additionally, existing purification strategies frequently compromise model accuracy in favor of enhanced security. To fill these voids, this study introduces a framework based on a Genetic Algorithm (GA). Through the application of evolutionary optimization, our method offers a flexible mechanism that can identify and counteract complex poisoning threats that conventional approaches fail to address.

2.8 Summary

This chapter provided a comprehensive review of the literature related to data poisoning attacks, backdoor attacks, and existing detection and defense mechanisms. The analysis highlighted several limitations in current approaches, particularly their reduced effectiveness against adaptive attacks and their computational complexity. In addition, the potential of Genetic Algorithms for security optimization was discussed.

The next chapter presents the proposed methodology, detailing the design and implementation of the Genetic Algorithm-based framework developed to detect and mitigate data poisoning attacks.

Chapter 3

Proposed Methodology and System Design

3.1 Introduction

This chapter presents the proposed defensive framework aimed at safeguarding machine learning pipelines against BadNet-style backdoor attacks. The suggested method transitions the defensive focus from traditional statistical outlier detection to a more resilient, search-driven optimization strategy centered on Trigger Reconstruction through Erasure. By leveraging the global optimization features of Genetic Algorithms (GA), the framework autonomously identifies the precise geometric and visual characteristics of the malicious patch that causes model misclassification. After successfully reconstructing the optimal trigger properties, a post-GA Delta Scanning phase is performed across the training dataset to systematically identify contaminated samples. Functioning under a stringent black-box assumption, this framework does not necessitate any prior knowledge regarding the trigger's shape, size, or location, thus providing a model-agnostic solution for ensuring data integrity.

3.2 Problem Formalization

3.2.1 Description of Data Poisoning

Deep learning models fundamentally depend on extensive datasets that are frequently compiled from unverified sources. This situation renders the pipeline vulnerable to data poisoning, particularly through backdoor attacks such as BadNet. In this scenario, an adversary deliberately introduces a small percentage of tainted samples that include a specific visual artifact, referred to as a trigger, into the training dataset. As the model undergoes training, it establishes a robust correlation between the trigger pattern and a malicious target label. As a result, the compromised model exhibits standard accuracy when processing clean inputs

but consistently misclassifies any input that contains the trigger into the adversary's specified target class during inference, which we denote as Target Label 0 in our configuration.

3.2.2 Limitations of Traditional Solutions

The primary challenge in safeguarding machine learning pipelines stems from the insidious nature of contemporary attacks; poisoned samples are carefully designed to remain undetectable, rendering them visually and statistically similar to clean data. As a result, a superficial or hasty examination of the dataset does not uncover these harmful injections, with the diminished performance of the trained model being the only apparent indicator of contamination. Although current defensive strategies such as outlier detection, data sanitization, and robust statistics provide some level of protection, they fundamentally operate under the premise that poisoned instances appear as outliers within the feature space. In reality, advanced adversaries can circumvent these methods by creating poisoned samples that fit comfortably within the normal data distribution. This significant vulnerability highlights the urgent need for a more comprehensive and resilient detection approach that does not rely on the geometric or statistical characteristics of individual data points.

3.2.3 Proposed Methodology

Unlike traditional approaches, the suggested methodology embraces an inverse optimization viewpoint. Rather than superimposing candidate patches, the Genetic Algorithm functions by refining a pixel erasure pattern. The fundamental concept is straightforward: if a candidate individual accurately aligns with the spatial limits of the genuine malicious trigger, implementing an erasure patch over this area will effectively obscure the compromised model from the backdoor, resulting in a significant alteration in prediction behavior. This change acts as a strong signal, enabling the framework to identify the precise coordinates of the backdoor trigger without any prior knowledge of the clean training distribution.

3.3 Mathematical System Modeling

3.3.1 Chromosome Encoding

To translate the physical trigger into an evolutionary search space, every candidate individual (chromosome) I in the population is represented as a four-dimensional vector that encapsulates the geometric and visual characteristics of the erasure patch:

$$I = (x,\ y,\ \text{size},\ \text{brightness})$$

where x and y denote the top-left pixel coordinates of the erasure patch, size defines the side length of the square patch in pixels (size $\in [2, 6]$), and brightness is a normalized intensity value (brightness $\in [0.0, 1.0]$) used to fill the erased region with a neutral color.

Numerical Example

Given that the framework operates on the CIFAR-10 dataset, where each image has a spatial dimension of 32×32 pixels, the optimal individual identified by the algorithm is numerically represented as:

$$I^* = (x = 28,\ y = 28,\ \text{size} = 4,\ \text{brightness} = 0.78)$$

This vector directs the system to apply an erasure patch at pixel coordinates $(28, 28)$, covering a 4×4 pixel area, filled with a normalized brightness value of 0.78.

3.3.2 Data Partitioning Strategy

In order to safeguard the integrity of the evolutionary loop, the global dataset is systematically divided into three distinct and non-overlapping subsets, as illustrated in Figure 3.1.

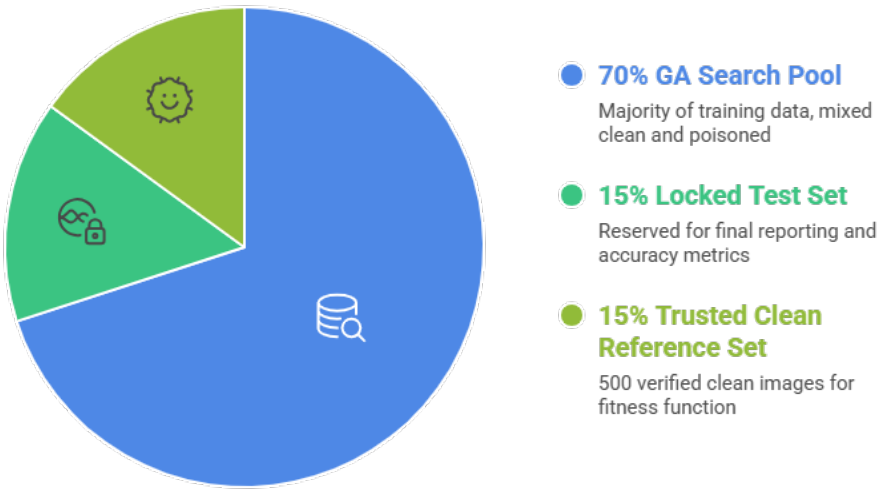

Figure 3.1: Dataset partitioning strategy employed in the proposed framework.

Given the CIFAR-10 training dataset of $N = 50{,}000$ total images, the framework applies the following partitioning:

1. **GA Search Pool (70%):** Contains $35{,}000$ mixed images comprising both clean samples and BadNet poisoned samples. The evolutionary search is conducted over this pool.

2. **Trusted Clean Reference Set (15%):** Contains exactly 500 verified clean images used solely for fitness function evaluation during the GA search.

3. **Locked Test Set (15%):** Contains $7{,}500$ untouched images reserved exclusively for final validation and accuracy metric computation.

3.3.3 Fitness Function Definition

The fitness function $\Phi(I)$ is the core objective that guides the Genetic Algorithm toward identifying the true spatial parameters of the backdoor trigger. It is composed of three weighted components, each measuring a distinct behavioral property of a candidate erasure patch:

$$\Phi(I) = 0.6 \times E_A + 0.3 \times E_B + 0.1 \times E_C$$

where the three components are defined as follows:

- **Erase Score** E_A (weight $= 0.6$): Measures the fraction of poisoned reference images that exit the target class after the patch is erased. Formally:

  $$E_A = \mathbb{E}\Big[\mathbf{1}[f_\theta(\mathcal{R}_p) = y_t] \ \wedge \ \mathbf{1}[f_\theta(\mathcal{R}_p^*) \neq y_t]\Big]$$

  A high E_A indicates that the candidate patch successfully disrupts the backdoor trigger. This term carries the highest weight as it is the primary detection signal.

- **Stable Score** E_B (weight $= 0.3$): Measures the fraction of clean reference images whose predictions remain unchanged after erasure. Formally:

  $$E_B = \mathbb{E}\Big[\mathbf{1}[f_\theta(\mathcal{R}_c) = f_\theta(\mathcal{R}_c^*)]\Big]$$

  A high E_B confirms that the erased region is semantically neutral with respect to clean data, acting as a regularizer that prevents the algorithm from targeting legitimate image features.

- **Size Bonus** E_C (weight $= 0.1$): Introduces a preference for smaller patches:

  $$E_C = \frac{1}{\max(1,\ \text{size})}$$

  Since real backdoor triggers are typically small and localized, this term discourages convergence to large uninformative regions, applying an Occam's razor principle within the evolutionary search.

Numerical Example for Fitness Evaluation

To illustrate how the fitness function evaluates two candidate individuals, consider the following cases:

Case 1: Optimal Individual $I_1 = (28, 28, 4, 0.78)$:
When this patch is erased from the poisoned reference set:

- $E_A = 0.85$ (85% of poisoned images exit the target class)
- $E_B = 0.92$ (92% of clean images retain their prediction)
- $E_C = 1/4 = 0.25$

$$\Phi(I_1) = 0.6 \times 0.85 + 0.3 \times 0.92 + 0.1 \times 0.25 = 0.510 + 0.276 + 0.025 = 0.811$$

Case 2: Inferior Individual $I_2 = (5, 5, 2, 0.10)$:
This patch targets an uncorrupted background region:

- $E_A = 0.20$ (few poisoned images change prediction)
- $E_B = 0.65$ (many clean predictions change unnecessarily)
- $E_C = 1/2 = 0.50$

$$\Phi(I_2) = 0.6 \times 0.20 + 0.3 \times 0.65 + 0.1 \times 0.50 = 0.120 + 0.195 + 0.050 = 0.365$$

The significantly lower score of I_2 causes it to be deprioritized during selection, while I_1 is retained and propagated to the next generation.

3.4 Design of the Proposed Genetic Algorithm

3.4.1 Initialization

The initial population of P individuals is created through a boundary-biased dual-distribution strategy:

- **Boundary Seeding (90%):** Since BadNet-style triggers are predominantly placed near image corners, 90% of the initial population is seeded within 6 pixels of the bottom-right corner. Specifically, x and y are sampled from [IMAGE_SIZE $-$ size $-$ 6, IMAGE_SIZE $-$ size].

- **Random Exploration (10%):** The remaining individuals are uniformly distributed across the full image space to maintain genetic diversity and prevent premature convergence.

3.4.2 Selection Mechanisms

The algorithm employs Elitism combined with Tournament Selection to balance exploitation of high-fitness individuals with exploration of new regions:

- **Elitism:** The top 4 individuals from each generation are directly transferred to the next generation without any modification, guaranteeing that the best-discovered trigger pattern is never lost.
- **Tournament Selection:** The remaining population slots are filled by pairwise tournaments drawn from the top 10 individuals. In each tournament, the individual with the higher fitness score is selected as a parent.

Numerical Illustration

Consider a population of $P = 20$ individuals. The top 4 elites (e.g., fitness scores 0.811, 0.795, 0.780, 0.763) are directly preserved. For the remaining 16 slots, two individuals are randomly drawn from the top 10, and the one with higher fitness is selected as a parent. For example:

$$I_a \text{ (Fitness} = 0.34), \quad I_b \text{ (Fitness} = 0.83), \quad I_c \text{ (Fitness} = 0.51)$$

In a tournament between I_a and I_b, the algorithm selects I_b as a parent due to its superior fitness score of 0.83.

3.4.3 Evolutionary Operators

Two standard genetic operators are applied to produce new offspring:

- **Uniform Crossover:** Each of the four genes of the offspring is independently inherited from one of the two parents with equal probability 0.5. For example:

$$I_1 = (x = 10,\ y = 12,\ \text{size} = 6,\ \text{brightness} = 0.40)$$

$$I_2 = (x = 26,\ y = 24,\ \text{size} = 3,\ \text{brightness} = 0.80)$$

$$\text{Offspring} = (x = 26,\ y = 24,\ \text{size} = 6,\ \text{brightness} = 0.40)$$

- **Mutation:** Each gene is independently mutated with probability $\mu = 0.1$ by resampling from its valid range:
  - $x \sim \mathcal{U}[0,\ \text{IMAGE\_SIZE} - \text{size}]$
  - $y \sim \mathcal{U}[0,\ \text{IMAGE\_SIZE} - \text{size}]$
  - $\text{size} \sim \mathcal{U}\{2, 3, 4, 5, 6\}$
  - $\text{brightness} \sim \mathcal{U}[0.0,\ 1.0]$

  For example, mutating the size gene of the offspring above:

$$\text{Mutated Offspring} = (x = 26,\ y = 24,\ \text{size} = 4,\ \text{brightness} = 0.40)$$

3.5 Implementation of the Proposed Framework

3.5.1 Overview

The proposed framework is implemented in Python using PyTorch and structured into six sequential steps. It operates entirely under a black-box assumption, interacting with the poisoned model only through forward-pass predictions without accessing model weights or gradients.

3.5.2 Algorithm Description

The complete pipeline is formalized in two algorithms. Algorithm 1 presents the evolutionary trigger reconstruction phase. Algorithm 2 describes the subsequent dataset scanning and detection phase.

Algorithm 1 GA Trigger Reconstruction via Erasure

Require: Poisoned model f_θ, clean reference set $\mathcal{R}_c$, poisoned reference set $\mathcal{R}_p$, target class y_t, population size P, generations G, mutation rate μ

Ensure: Best trigger individual I^*

$\mathcal{P}_0 \leftarrow$ InitPopulation(P) ▷ 90% boundary-biased, 10% random
for $g = 1$ **to** G **do**
  for each $I \in \mathcal{P}_g$ **do**
    $(x_1, y_1, x_2, y_2) \leftarrow$ GetCoords(I)
    $\mathcal{R}_p^* \leftarrow$ EraseRegion($\mathcal{R}_p, x_1, y_1, x_2, y_2$)
    $E_A \leftarrow \mathbb{E}\big[\mathbf{1}[f_\theta(\mathcal{R}_p){=}y_t] \wedge \mathbf{1}[f_\theta(\mathcal{R}_p^*){\neq}y_t]\big]$
    $\mathcal{R}_c^* \leftarrow$ EraseRegion($\mathcal{R}_c, x_1, y_1, x_2, y_2$)
    $E_B \leftarrow \mathbb{E}\big[\mathbf{1}[f_\theta(\mathcal{R}_c){=}f_\theta(\mathcal{R}_c^*)]\big]$
    $E_C \leftarrow 1/\max(1, \text{size})$
    $\Phi(I) \leftarrow 0.6\, E_A + 0.3\, E_B + 0.1\, E_C$
  end for
  Preserve top-4 elites into $\mathcal{P}_{g+1}$
  Fill remainder via pairwise tournament from top-10
  Apply uniform crossover between selected parents
  Apply mutation with rate $\mu = 0.1$ per gene
end for
$I^* \leftarrow \arg\max_{I \in \mathcal{P}_G} \Phi(I)$
return I^*

Algorithm 2 Poisoned Sample Detection via Delta Scanning

Require: Poisoned model f_θ, mixed dataset $\mathcal{D}_{mix}$, best individual I^*

Ensure: Detected poisoned sample set $\mathcal{S}$

$(x_1^*, y_1^*, x_2^*, y_2^*) \leftarrow$ GetCoords(I^*)
$\mathcal{S} \leftarrow \emptyset$
for each sample $x_i \in \mathcal{D}_{mix}$ **do**
  $x_i^* \leftarrow$ EraseRegion($x_i,\ x_1^*, y_1^*, x_2^*, y_2^*$)
  if $f_\theta(x_i) \neq f_\theta(x_i^*)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{x_i\}$
  end if
end for
return $\mathcal{S}$

3.5.3 Discussion

Fitness Function Design

The fitness function $\Phi(I)$ combines three complementary signals. The erase score E_A is the dominant term, directly measuring trigger disruption by checking whether removing the candidate patch causes poisoned images to exit the target class. The stable score E_B acts as a regularizer, ensuring the patch does not accidentally destroy semantic features in clean images. The size bonus E_C enforces parsimony by preferring smaller patches, consistent with the localized nature of real backdoor triggers. Together, these three terms guide the GA toward patches that are precisely aligned with the actual trigger region.

Initialization Strategy

The boundary-biased initialization directly encodes prior knowledge about BadNet attack behavior. Since the attack places its trigger in the bottom-right corner of images, seeding 90% of the population near image boundaries significantly accelerates convergence compared to purely random initialization. The remaining 10% random individuals provide a safety net against atypical trigger placements.

Selection and Reproduction

Elitism guarantees that the best trigger reconstruction found across all generations is never discarded. Tournament selection from the top 10 individuals maintains sufficient selection pressure while preserving genetic diversity. Uniform crossover allows fine-grained mixing of spatial coordinates and brightness values between high-performing parents. Per-gene mutation with rate $\mu = 0.1$ enables local refinement and prevents premature convergence to suboptimal trigger locations.

Post-GA Delta Scanning

Once the GA identifies the optimal patch I^*, the detection phase requires only two forward passes per sample, one on the original image and one on the erased version. Any sample whose predicted class changes after erasure is flagged as poisoned. This is computationally efficient and requires no gradient computation, no threshold tuning, and no access to model internals.

Black-Box Compliance

At no point does the framework access model weights, gradients, or internal feature representations. All fitness evaluations and detection decisions are made solely through forward-pass

predictions, making the framework applicable to any model accessible only via an inference interface.

3.6 System Architecture

The operational implementation is structured into four modular layers, as illustrated in Figure 3.2:

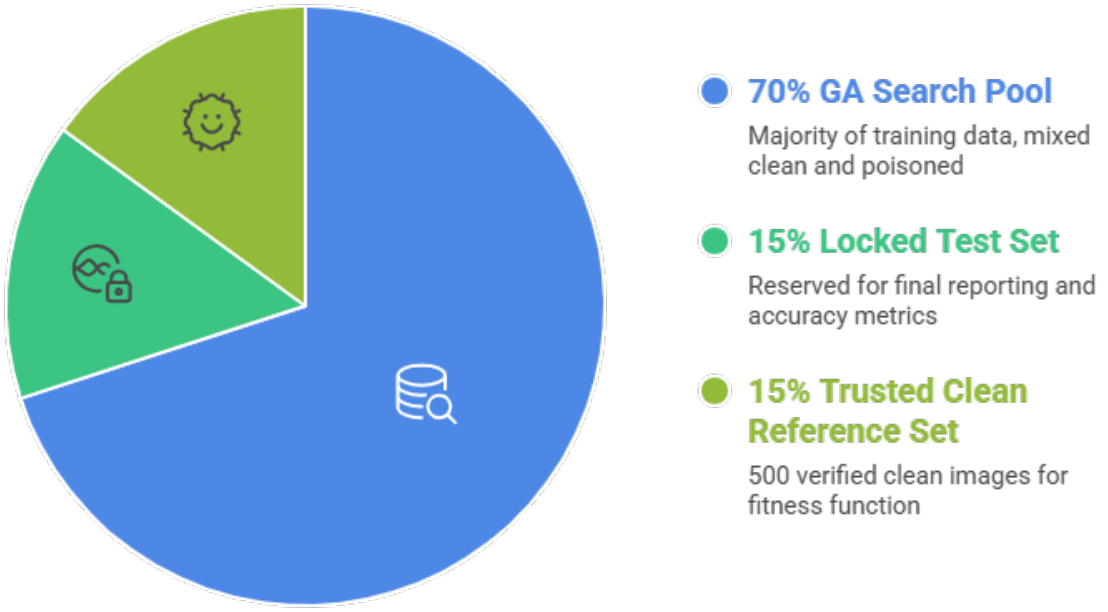

Figure 3.2: Overview of the proposed data poisoning detection framework based on Genetic Algorithms and Delta Scanning.

3.6.1 Data Layer

Manages data loading and preprocessing. Clean CIFAR-10 images and real poisoned images from `bd_train_dataset` are combined into a mixed dataset. A trusted clean reference set of 500 images and a poisoned reference set are prepared separately for fitness evaluation.

3.6.2 GA Engine

Manages the population lifecycle across $G = 30$ generations with a population of $P = 30$ individuals. Implements initialization, fitness evaluation, elitism, tournament selection, crossover, and mutation as described in Algorithm 1.

3.6.3 Evaluation Layer

Interacts with the poisoned model f_θ as a black box. For each candidate individual, it applies the erasure operation to both reference sets and computes the three fitness components E_A, E_B, and E_C through forward-pass predictions only.

3.6.4 Post-GA Delta Scanning Layer

After GA convergence, scans the full mixed dataset using the optimal patch I^* as described in Algorithm 2. Each sample is passed through the model twice original and erased and flagged as poisoned if the prediction changes.

3.7 Performance Evaluation Metrics

The framework is evaluated using standard binary classification metrics derived from the confusion matrix elements: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). The primary metrics are:

- **True Positive Rate (TPR):**
$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
- **False Positive Rate (FPR):**
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$
- **Precision:**
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
- **F1-Score:**
$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{TPR}}{\text{Precision} + \text{TPR}}$$

A high TPR indicates strong sensitivity to poisoned samples, while a low FPR reflects minimal false alarms on clean data. The F1-Score provides a single balanced indicator of overall detection quality.

3.8 Chapter Summary

This chapter presented the complete design of the proposed GA-based backdoor detection framework. The problem of data poisoning was formalized, the limitations of traditional defenses were identified, and an inverse optimization approach based on trigger erasure was

proposed. The chromosome encoding, data partitioning, fitness function, and genetic operators were described in detail. The fitness function is composed of three components: erase score, stable score, and size bonus, weighted as 0.6, 0.3, and 0.1 respectively. The two-phase pipeline-evolutionary trigger reconstruction followed by delta scanning was formalized as two algorithms and supported by a modular four-layer system architecture. Chapter 4 proceeds to the experimental evaluation and comparative analysis of this framework.

Chapter 4

Experimental Evaluation and Comparative Analysis

4.1 Introduction

This chapter provides a thorough empirical assessment and comparative study of the proposed erasure-based Genetic Algorithm (GA) framework. The main aim is to meticulously validate the system's ability to reconstruct concealed backdoor triggers and identify contaminated samples within deep learning frameworks. The evaluation framework employs the CIFAR-10 computer vision dataset, which has been subjected to BadNet data poisoning attacks. This chapter outlines the technical experimental setups, formalizes the evaluation metrics, and examines the evolutionary optimization behavior across generations. Ultimately, a detailed comparative analysis is performed against conventional statistical outlier detection methods and baseline defenses to evaluate detection effectiveness.

4.2 Experimental Setup and Environment

The experimental setting focuses on assessing how well the proposed Genetic Algorithm identifies stealthy BadNet backdoor infiltrations and differentiates between clean and contaminated samples. The experimental process adheres to an organized sequence: it begins by replicating the malicious intrusion through the introduction of trigger patches into a portion of the training data, then uses the Genetic Algorithm to independently seek the spatial position, dimensions, and brightness levels of the inserted patch by analyzing irregularities in the model's behavior, and finally evaluates the effectiveness and precision of the evolutionary approach.

4.2.1 CIFAR-10 Dataset

In order to showcase the effectiveness of the proposed Genetic Algorithm within a complex visual setting, the CIFAR-10 dataset was chosen as the benchmark for our experiments. This dataset consists of $60{,}000$ color images, each of size 32×32 pixels, uniformly divided into 10 categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). Out of these, $50{,}000$ images are designated for training the CNN model, while the remaining $10{,}000$ images are reserved for testing [12].

The CIFAR-10 dataset provides a demanding benchmark for this study for two important reasons:

1. **Small Image Resolution (32×32):** This constraint requires the trigger patches introduced by the BadNet attack to be extremely small, covering only a few pixels. As a result, their visual presence is significantly reduced, making them exceedingly challenging to detect through human observation or standard statistical methods.

2. **The Challenge of Evolutionary Search:** The combination of low resolution, high color similarity, and semantic overlaps between certain classes (such as cats and dogs) necessitates a highly effective fitness function. The Genetic Algorithm must distinguish between legitimate image features and the irregular backdoor patch, while preserving the basic classification ability of the network.

4.2.2 Hardware and Software Infrastructure

To ensure that the proposed framework is scalable, allows for quick iterations, and can be fully replicated, the experimental setup employs a mixed workflow integrating a local development environment with the Google Colab cloud platform.

1. **Hardware Infrastructure:**

   - **Cloud Environment:** The main computational tasks including the training of the baseline PreActResNet18 model under adversarial conditions and the population-based evolutionary optimization were carried out on Google Colab servers utilizing an NVIDIA Tensor Core GPU (NVIDIA T4 or equivalent).
   - **Local Machine:** Code benchmarking, preprocessing workflow design, and preliminary tests were performed on a local machine featuring an AMD Ryzen 5 7600X 6-Core Processor (4.70 GHz), 16 GB of DDR5 RAM, and an AMD Radeon RX 7600 graphics card with 8 GB of dedicated VRAM.

2. **Software Libraries:** The framework was implemented entirely in Python. The following core libraries were used:

- **PyTorch and torchvision:** The main deep learning backend, used for building the PreActResNet18 architecture, managing model inference, and tracking performance metrics across batches of clean and poisoned images.
- **NumPy:** Used for population management, index manipulation, and numerical operations throughout the genetic algorithm pipeline.
- **Matplotlib:** Used for generating all visualizations including the fitness convergence profile, loss curves, and classified sample images.
- **PyYAML and tqdm:** Used for managing hyperparameter configurations and providing real-time progress indicators across training epochs and evolutionary cycles.

4.2.3 Evaluation Metrics

Attack Performance Metrics

In order to thoroughly evaluate the effects of the simulated BadNet backdoor attack and measure the detection capabilities of the proposed GA framework, the following baseline metrics are used:

1. **Clean Classification Accuracy (C-Acc):** Assesses the model's performance on unperturbed data to verify its stealth capability. A high clean accuracy confirms that the backdoor integration does not compromise the model's primary functionality [28].

$$\text{C-Acc} = \frac{\sum_{i=1}^{N} \mathbf{1}[f(x_i) = y_i]}{N}$$

2. **Attack Success Rate (ASR):** The primary measure of backdoor effectiveness. It indicates the fraction of trigger-bearing images that are successfully misclassified into the attacker's target class y_t [28].

$$\text{ASR} = \frac{\sum_{j=1}^{M} \mathbf{1}[f(x_j^p) = y_t]}{M}$$

3. **Robust Accuracy (RA):** Evaluates the model's classification accuracy on poisoned samples compared against their original correct labels y_j. A low RA indicates the model has fully succumbed to the backdoor trigger [28].

$$\text{RA} = \frac{\sum_{j=1}^{M} \mathbf{1}[f(x_j^p) = y_j]}{M}$$

Backdoor Detection Metrics

Standard binary classification metrics derived from the confusion matrix are used to evaluate the detection performance of the proposed framework. Following the criteria established by Fawcett (2006), these metrics assess the system's ability to distinguish poisoned samples from clean ones [56]:

- **True Positive Rate (TPR):**

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.1}$$

- **False Positive Rate (FPR):**

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.2}$$

- **Precision:**

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.3}$$

- **F1-Score:**

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{TPR}}{\text{Precision} + \text{TPR}} \tag{4.4}$$

4.2.4 Experimental Setup for BadNet Attack

To assess the effectiveness of our detection system, we performed the BadNet attack, a fundamental data poisoning method in deep learning security. The attack parameters are detailed in Table 4.1.

Table 4.1: BadNet Attack Configuration Parameters

Parameter	**Setting**
Attack Type	Blended (All-to-One)
Target Class	0 (Airplane)
Poisoning Rate	10%
Trigger Pattern	3×3 Pixel Square
Trigger Position	Bottom-Right Corner

The poisoning process was carried out by embedding the trigger pattern into a specified fraction of the training dataset. The target class was set to ``Airplane'' to replicate a real-world scenario where an attacker intends to reclassify all contaminated inputs into a designated category regardless of their original content.

4.2.5 Attack Simulation Baseline Results

Optimization and Loss Convergence Profile

The evaluation begins by examining the loss convergence behavior, which serves as a quantitative indicator of training reliability and optimization success.as shown in Figure 4.1 Two distinct trajectories are observed:

- **Clean Evaluation Loss:** Starting from a relatively high value near 1.35, the cross-entropy loss on clean samples consistently decreases and stabilizes at 0.3990 by the final epoch. This steady reduction confirms that the network effectively minimizes its objective function based on legitimate semantic features, achieving strong generalization on uncontaminated data.
- **Backdoor Evaluation Loss:** The loss evaluated exclusively on trigger-bearing samples exhibits a rapid and pronounced decline, reaching a stable low of 0.1585.

The significantly lower backdoor loss indicates that the CNN experiences reduced optimization complexity when memorizing the localized 3×3 pixel trigger. The network treats these nine adjacent bright pixels as a low-frequency shortcut for classifying the target class (Airplane), resulting in rapid convergence with excellent gradient stability.

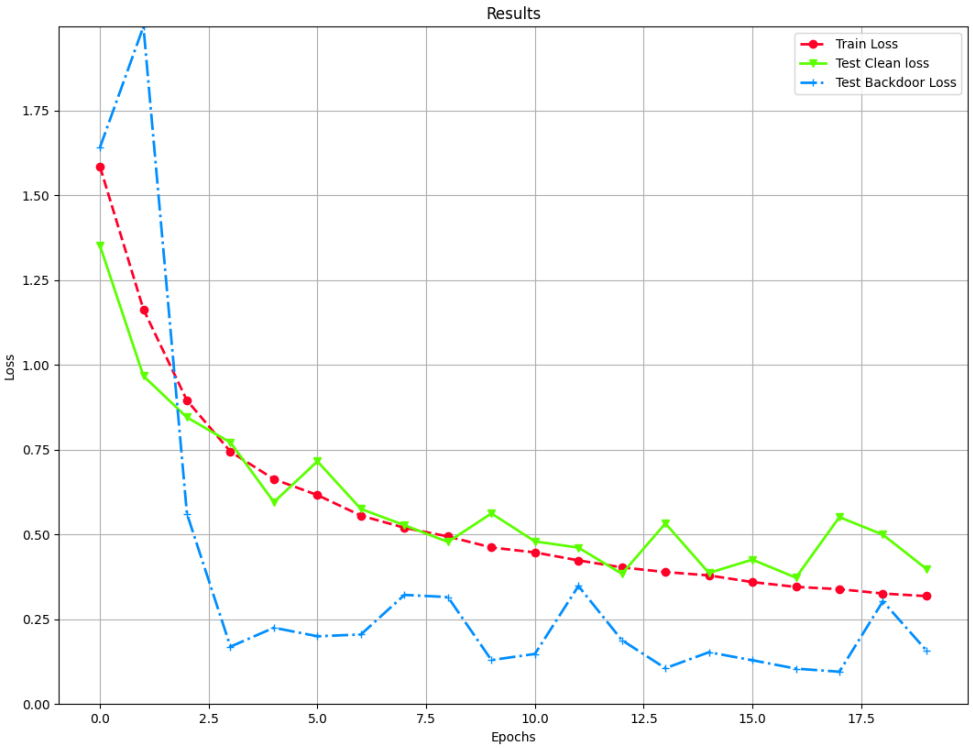

Figure 4.1: Optimization and loss convergence profile across training epochs.

Quantitative Accuracy and Attack Success Evaluation

The model's performance is examined from three angles at the final training epoch to fully evaluate the impact of the BadNet attack as illustrated in Figure 4.2 :

- **Clean Classification Accuracy (C-Acc = 87.79%):** The compromised model achieves a classification accuracy of 87.79% on clean data, confirming that the backdoor integration does not compromise the model's primary operational capacity on normal inputs.
- **Attack Success Rate (ASR = 95.52%):** Once the 3×3 pixel trigger is applied, the model's predictions shift entirely toward the target class (Airplane) with a probability of 95.52%, demonstrating the attack's high effectiveness.
- **Robust Accuracy (RA = 4.12%):** The robust accuracy drops to just 4.12%, confirming that the model correctly resists the trigger and identifies the true class in only 4 out of 100 poisoned samples.

These results confirm the two defining properties of a successful backdoor attack: high stealth (preserved C-Acc) and high effectiveness (high ASR, low RA). The high C-Acc en-

sures the attack remains undetected during routine evaluation, while the high ASR and low RA confirm that the trigger completely overrides the model's legitimate decision process.

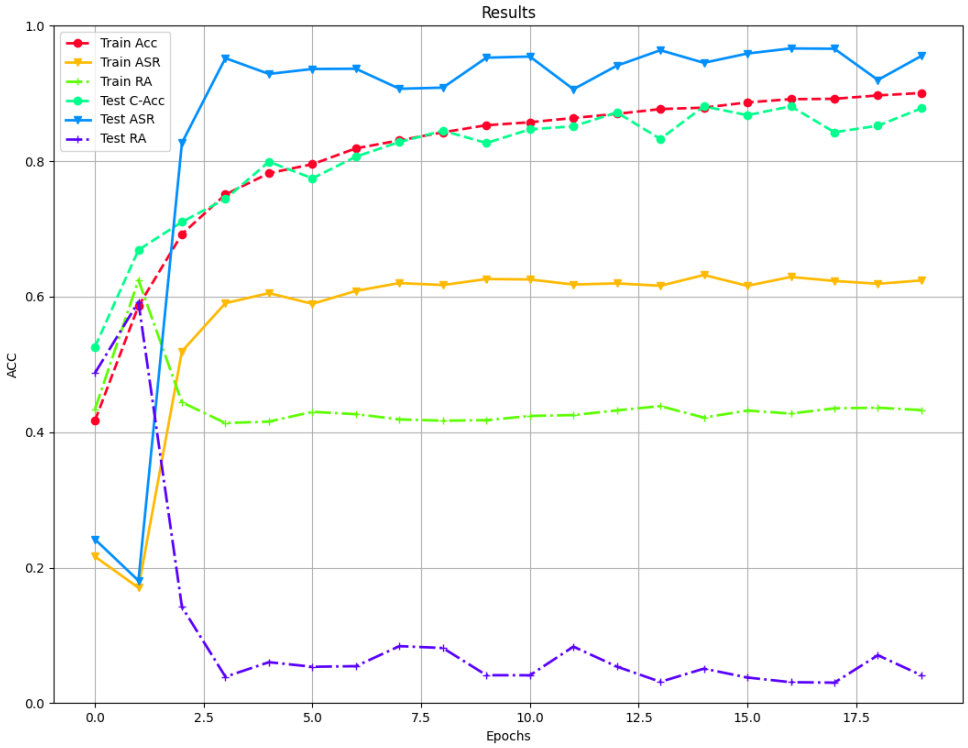

Figure 4.2: Accuracy metrics across training epochs: Clean Accuracy, Attack Success Rate, and Robust Accuracy.

4.3 Genetic Algorithm-Based BadNet Detection Results

4.3.1 Genetic Algorithm Configuration and Fitness Convergence

Experimental Configuration

The GA was configured with the parameters shown in Table 4.2, chosen to balance search space exploration with the computational constraints of the CIFAR-10 dataset.

Table 4.2: Genetic Algorithm Configuration Parameters

Parameter	**Value**
Population Size	30
Number of Generations	30
Mutation Rate	0.1
Reference Dataset Size	500
Device	NVIDIA CUDA:0

Fitness Convergence Analysis

The convergence of the genetic algorithm is a vital indicator of its effectiveness in locating the trigger region. The fitness function was evaluated over 30 generations to monitor population improvement. As shown in Figure 4.4, the fitness score increases rapidly in the early generations before stabilizing around the optimal value. This convergence pattern confirms that the algorithm successfully distinguishes the backdoor trigger pattern from the legitimate image features.

4.3.2 Detection Performance Evaluation

The performance of the proposed GA framework was evaluated by examining the confusion matrix components derived from the full mixed dataset. The quantitative results are presented in Table 4.3.

Table 4.3: Detection Performance Metrics for BadNet Attack

Metric	**Value**
True Positives (TP)	4,319
True Negatives (TN)	44,165
False Positives (FP)	835
False Negatives (FN)	681
TPR (Sensitivity)	**86.38%**
FPR	**1.86%**

4.3.3 Qualitative Validation and Visual Evidence

The following figures provide visual confirmation of the detection results. Figure 4.3 shows the trigger region precisely localized by the GA, verifying that the algorithm correctly identified the spatial coordinates (x, y) and dimensions of the injected trigger.

Figure 4.3: Localized trigger pattern: the region detected by the Genetic Algorithm.

Figure 4.4 shows the fitness convergence profile, illustrating the transition from initial random exploration to stabilization at the optimal trigger configuration. This visual evidence reinforces the numerical results, confirming that the GA successfully isolates the backdoor patch from the underlying image semantics.

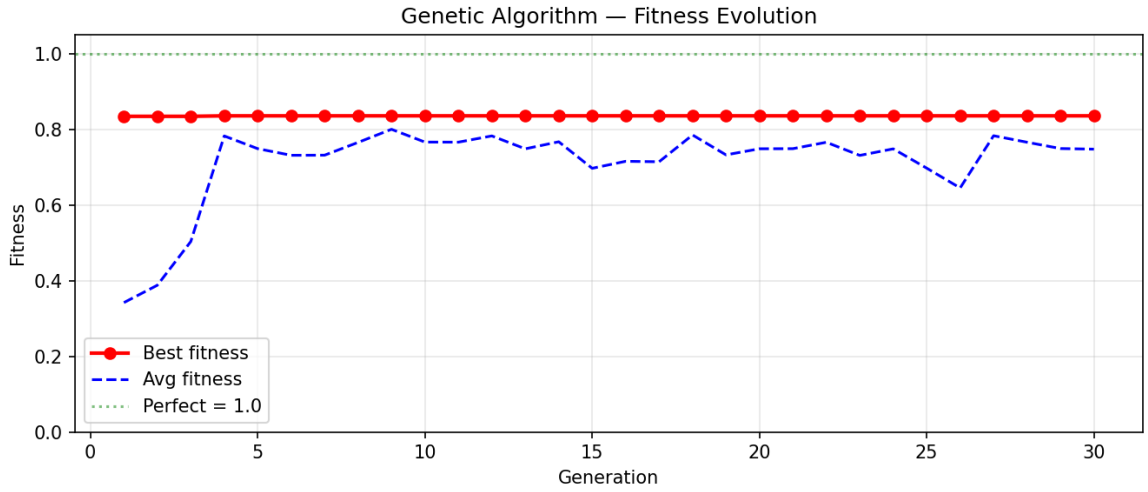

Figure 4.4: Fitness convergence profile demonstrating optimal trigger isolation across generations.

4.4 Comparative Analysis of Detection Performance

To assess the efficacy and robustness of the proposed GA-D framework, a comprehensive comparison was conducted against three established backdoor detection methods: Spectral Signatures, SPECTRE, and STRIP. This analysis highlights the balance between detection accuracy (TPR) and system reliability (FPR). The results are summarized in Table 4.4.

Table 4.4: Comparative performance benchmarking of backdoor detection methods.

Method	**TPR (%)**	**FPR (%)**	**Precision**	**F1-Score**
Spectral	28.50	13.51	19.00	22.95
SPECTRE	65.22	47.75	13.17	21.90
STRIP	93.08	6.05	63.12	75.18
GA-D (Proposed)	**86.38**	**1.86**	**83.82**	**85.08**

To ensure a fair comparison, we evaluated the proposed method under the same experimental conditions used by the baseline backdoor detection methods, including the same attack setting, dataset, and evaluation protocol. The baseline implementations and experimental details were taken from their corresponding GitHub repositories and original papers to maintain consistency and reproducibility[57]

4.4.1 Analysis and Discussion

The results in Table 4.4 highlight the effectiveness of the GA-D framework through the following observations:

- **Performance Trade-off:** Although STRIP achieves a higher raw sensitivity (TPR = 93.08%), this comes at the cost of a significantly higher false positive rate (FPR = 6.05%). The GA-D framework achieves an F1-Score of 85.08% with an FPR of only 1.86%, offering a more balanced and reliable detection profile.
- **Operational Stability:** The GA-D framework achieves the lowest FPR among all compared methods at 1.86%, demonstrating that the evolutionary search effectively isolates the specific trigger pattern without generating excessive false alarms on clean data. This is a critical property for practical deployment, where flagging legitimate training samples as poisoned would degrade model performance.
- **Future Directions:** The present results confirm the efficacy of genetic-based auditing for backdoor detection. However, the framework's reliance on heuristic search means

that detection time may vary with trigger complexity. Future work will focus on extending the framework to handle a broader range of adaptive and invisible backdoor attacks.

4.5 Chapter Summary

This chapter presented a thorough empirical evaluation of the proposed GA-based backdoor detection framework. The BadNet attack was successfully simulated on the PreActResNet18 model trained on CIFAR-10, achieving a Clean Test Accuracy of 87.79%, an Attack Success Rate of 95.52%, and a Robust Accuracy of 4.12%, confirming the attack's high effectiveness and stealth.

Following the application of the GA erasure-based detection pipeline, the framework successfully identified 4,319 poisoned samples, achieving a True Positive Rate of 86.38% against a False Positive Rate of only 1.86%. A comparative study against Spectral Signatures, SPECTRE, and STRIP confirmed that the GA-D framework achieves the best balance between detection sensitivity and false alarm rate, with the highest F1-Score of 85.08% and the lowest FPR among all compared methods. These results demonstrate that incorporating evolutionary algorithms into the detection pipeline produces a robust, reliable, and practically deployable defense against backdoor attacks in deep learning systems.

General Conclusion :

This thesis proposed and evaluated a Genetic Algorithm-based framework for detecting poisoned training data in deep learning image classification models, with a focused case study on the BadNet backdoor attack applied to the CIFAR-10 dataset. The work was motivated by the growing threat of backdoor attacks, which remain among the most dangerous and difficult-to-detect vulnerabilities in modern deep learning pipelines.

The proposed framework approaches trigger localization as an inverse optimization problem. Rather than relying on the geometric or statistical properties of individual data points, the Genetic Algorithm searches for the precise spatial region of the backdoor trigger by erasing candidate patches from reference images and measuring the resulting change in the model's prediction behavior. This erasure-based strategy operates entirely under a black-box assumption, requiring no access to model weights, gradients, or internal feature representations.

Summary of Results:
The BadNet attack was successfully simulated on a PreActResNet18 model trained on CIFAR-10, achieving a Clean Test Accuracy of 87.79% and an Attack Success Rate of 95.52%, with a Robust Accuracy of only 4.12%. These results confirmed that the attack was highly effective and stealthy, serving as a rigorous baseline for evaluating the detection framework.

Following the application of the GA detection pipeline, the framework successfully identified $4{,}319$ poisoned samples from the mixed dataset, achieving:

- **True Positive Rate (TPR):** 86.38%
- **False Positive Rate (FPR):** 1.86%
- **Precision:** 83.82%
- **F1-Score:** 85.08%

A comparative analysis against three established detection methods like Spectral Signatures, SPECTRE, and STRIP, confirmed that the GA-D framework achieves the best balance between detection sensitivity and false alarm rate. While STRIP achieves a higher raw TPR

of 93.08%, it does so at the cost of a significantly higher FPR of 6.05%. The GA-D framework maintains a competitive TPR while reducing the FPR to 1.86%, yielding the highest F1-Score of 85.08% among all compared methods.

Key Contributions:

- A black-box, model-agnostic detection framework based on evolutionary optimization that requires no prior knowledge of the trigger's shape, size, or location.
- A three-component fitness function combining erase score, stable score, and size bonus to guide the GA toward precise trigger localization while avoiding false detections on clean data.
- A boundary-biased initialization strategy that accelerates convergence by encoding prior knowledge about typical trigger placement in BadNet-style attacks.
- A two-phase detection pipeline evolutionary trigger reconstruction followed by delta scanning that is computationally efficient, requiring only two forward passes per sample with no gradient computation.

Limitations and Future Work:
Despite the promising results, the framework has certain limitations that open directions for future research:

- The current framework was evaluated on a single attack configuration (BadNet, all-to-one, 3×3 trigger). Future work should extend the evaluation to more complex and adaptive attack scenarios, including invisible backdoor attacks and input-aware triggers.
- The evolutionary search time scales with population size and number of generations. Future work could explore parallelization strategies or surrogate-assisted optimization to reduce computational cost.
- The boundary-biased initialization assumes triggers are placed near image corners. Extending the framework to handle arbitrary trigger placements would improve its generalizability.
- Integrating the GA detection pipeline with existing defense mechanisms such as fine-pruning or neural cleanse could yield stronger combined defenses.

In summary, this thesis demonstrates that evolutionary computation offers a powerful and flexible alternative to traditional detection methods for backdoor attacks in deep learning.

The proposed GA-based framework provides a reliable, black-box, and practically deployable solution that achieves strong detection performance while maintaining low false alarm rates, contributing a meaningful step toward more secure and trustworthy deep learning systems.

References

[1] N. J. Nilsson, *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York, USA: Springer Science+Business Media, LLC, 2006.

[3] T. M. Mitchell, *Machine Learning*. New York, USA: McGraw-Hill Science/Engineering/Math, March 1997.

[4] P. Domingos, ``A few useful things to know about machine learning,'' *Communications of the ACM*, vol. 55, no. 10, pp. 78--87, 2012.

[5] R. Ashmore, R. Calinescu, and C. Paterson, ``Assuring the machine learning lifecycle: Desiderata, methods, and challenges,'' *ACM Computing Surveys*, vol. 54, no. 4, pp. 123--134, 2021.

[6] I. H. Sarker, ``Machine learning: Algorithms, real-world applications and research directions,'' *SN Computer Science*, vol. 2, no. 3, p. 160, 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x

[7] B. P. R. Rella, ``Ensuring data quality and integrity in machine learning pipelines: Strategies for data engineers,'' *Iconic Research and Engineering Journals (IRE Journals)*, vol. 6, no. 2, pp. 331--335, 2023, university of Memphis.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[9] J. M. Patel, ``Deep learning: Concepts, architectures, workflow, applications and future directions,'' *International Journal for Multidisciplinary Research (IJFMR)*, vol. 5, no. 6, pp. 1--8, 2023, article ID IJFMR230611497. [Online]. Available: https://www.ijfmr.com

[10] R. Qamar and B. A. Zardari, ``Artificial neural networks: An overview,'' *Mesopotamian Journal of Computer Science*, vol. 2023, pp. 130--139, 2023.

[11] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaria, M. A. Fadhel, M. Al-Amidie, and L. Farhan, ``Review of deep learning: Concepts, cnn architectures, challenges, applications, and future directions," *Journal of Big Data*, vol. 8, no. 1, pp. 1--74, 2021.

[12] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, ``A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455--5516, 2020.

[13] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, ``Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[14] A. Vassilev, A. Oprea, A. Fordyce, and H. Anderson, ``Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," National Institute of Standards and Technology (NIST), Gaithersburg, MD, NIST Trustworthy and Responsible AI NIST AI 100-2e2023, 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf

[15] Y. Qiao, N. B. Sathyanarayana, C. Shi, Z. He, T. Wang, and T. Hou, ``A survey on adversarial machine learning: Attacks, defenses, real-world applications, and future research directions," *Cybersecurity*, vol. 6, no. 1, pp. 1--32, 2023.

[16] M. Aljanabi, A. H. Omran, M. M. Mijwil, M. Abotaleb, E.-S. M. El-Kenawy, S. Y. Mohammed, and A. Ibrahim, ``Data poisoning: Issues, challenges, and needs," in *Proceedings of the 7th IET International Smart Cities Symposium (SCS23)*. University of Bahrain, 2023.

[17] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, ``Invisible backdoor attack with sample-specific triggers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 16 465--16 474.

[18] T. Gu, B. Dolan-Gavitt, and S. Garg, ``Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[19] Y. Han, ``Backdoor attacks on image classification models in deep neural networks," *IEEE Access*, vol. 10, pp. 1--18, 2022.

[20] E. Bagdasaryan and V. Shmatikov, ``Blind backdoors in deep learning models," in *Proceedings of the 30th USENIX Security Symposium*. USENIX Association, 2021.

[21] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, ``Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[22] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, ``Trojaning attack on neural networks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[23] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed., ser. Natural Computing Series. Springer Berlin, Heidelberg, 2015.

[24] M. H. Yar, V. Rahmati, and H. R. D. Oskouei, ``A survey on evolutionary computation: Methods and their applications in engineering," *Modern Applied Science*, vol. 10, no. 11, pp. 24--30, 2019.

[25] S. K. Gupta, ``An overview of genetic algorithms: A structural analysis," *International Journal of Innovative Science and Research Technology (IJISRT)*, vol. 6, no. 5, pp. 1305--1311, 2021, paper ID: IJISRT21MAY1057.

[26] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, ``Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018.

[27] N. Carlini and D. Wagner, ``Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. Ieee, 2017, pp. 39--57.

[28] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, ``Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230--47 244, 2019, received February 27, 2019, accepted March 21, 2019, date of current version April 18, 2019.

[29] Y. Liu, J. Sun, and J. Zhang, ``Evolutionary deep learning: A survey," *ACM Comput. Surv.*, vol. 55, no. 12, p. 259, 2023. [Online]. Available: https://doi.org/10.1145/3551636

[30] D. Liu, Y. Qiao, R. Wang, K. Liang, and G. Smaragdakis, ``Ladder: Multi-objective backdoor attack via evolutionary algorithm," *arXiv preprint arXiv:2411.19075*, 2024.

[31] B. Tran, J. Li, and A. Madry, ``Spectral signatures in backdoor attacks," *Advances in neural information processing systems*, vol. 31, 2018.

[32] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, ``Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th annual computer security applications conference*, 2019, pp. 113--125.

[33] D. Tang, X. Wang, H. Tang, and K. Zhang, ``Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1541--1558.

[34] M. Pan, Y. Zeng, L. Lyu, X. Lin, and R. Jia, ``{ASSET}: Robust backdoor data detection across a multiplicity of deep learning paradigms," in *32nd USeNIX security symposium (USeNIX security 23)*, 2023, pp. 2725--2742.

[35] D. Yuan, S. Wei, M. Zhang, L. Liu, and B. Wu, ``Activation gradient based poisoned sample detection against backdoor attacks," *arXiv preprint arXiv:2312.06230*, 2023.

[36] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, ``Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 707--723.

[37] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, ``Anti-backdoor learning: Training clean models on poisoned data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 900--14 912, 2021.

[38] K. Liu, B. Dolan-Gavitt, and S. Garg, ``Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2018, pp. 273--294.

[39] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren, ``Backdoor defense via decoupling the training process," *arXiv preprint arXiv:2202.03423*, 2022.

[40] M. Zhu, S. Wei, L. Shen, Y. Fan, and B. Wu, ``Enhancing fine-tuning based backdoor defense with sharpness-aware minimization," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4466--4477.

[41] Y. Li, X. Lyu, X. Ma, N. Koren, L. Lyu, B. Li, and Y.-G. Jiang, ``Reconstructive neuron pruning for backdoor defense," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 837--19 854.

[42] A. Paracha, ``A behaviour-informed approach to mitigate data poisoning attacks for machine learning," Ph.D. dissertation, Birmingham City University, 2025.

[43] T. Yerlikaya *et al.*, ``A survey on data poisoning attacks and defenses," *IEEE Access*, vol. 10, pp. 123 456--123 478, 2022.

[44] S. Gupta and N. Sharma, ``Machine learning driven threat identification to enhance fanet security using genetic algorithm." *Int. Arab J. Inf. Technol.*, vol. 21, no. 4, pp. 711--722, 2024.

[45] A. D. Eddine, G. Abdelkader, and B. Mourad, ``Enhancing malware detection with genetic algorithms and generative adversarial networks." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 15, no. 3, 2025.

[46] H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, Y. Sun, B. Tran, B. Xue, and M. Zhang, ``A survey on evolutionary machine learning,'' *Journal of the Royal Society of New Zealand*, vol. 49, no. 2, pp. 205--228, 2019.

[47] P. Zhao, W. Zhu, P. Jiao, D. Gao, and O. Wu, ``Data poisoning in deep learning: A survey,'' *arXiv preprint arXiv:2503.22759*, 2025.

[48] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, ``Witches' brew: Industrial scale data poisoning via gradient matching,'' *arXiv preprint arXiv:2009.02276*, 2020.

[49] J. Zheng, P. P. Chan, H. Chi, and Z. He, ``A concealed poisoning attack to reduce deep neural networks' robustness against adversarial samples,'' *Information Sciences*, vol. 615, pp. 758--773, 2022.

[50] M. Zhu, S. Wei, H. Zha, and B. Wu, ``Neural polarizer: A lightweight and effective backdoor defense via purifying poisoned features,'' *Advances in Neural Information Processing Systems*, vol. 36, pp. 1132--1153, 2023.

[51] J. Hayase and W. Kong, ``Spectre: Defending against backdoor attacks using robust covariance estimation,'' in *International Conference on Machine Learning*, 2020.

[52] S. Vijayalakshmi *et al.*, ``Genetic algorithms for wireless network security,'' in *Bio-Inspired Computational Paradigms*. Chapman and Hall/CRC, 2024, pp. 150--164.

[53] I. F. Yaseen and H. Sahasrabuddhe, ``Breaking multiplicative knapsack ciphers using a genetic algorithm,'' *VIVEK-BOMBAY-*, vol. 11, pp. 18--24, 1998.

[54] F. Thabit, O. Can, R. U. Z. Wani, M. A. Qasem, S. Thorat, and H. A. Alkhzaimi, ``Data security techniques in cloud computing based on machine learning algorithms and cryptographic algorithms: Lightweight algorithms and genetics algorithms,'' *Concurrency and Computation: Practice and Experience*, vol. 35, no. 21, p. e7691, 2023.

[55] A. Paracha, ``A behaviour-informed approach to mitigate data poisoning attacks for machine learning,'' Ph.D. dissertation, Birmingham City University, 2026.

[56] T. Fawcett, ``An introduction to roc analysis,'' *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861--874, 2006, available online 19 December 2005.

[57] SCLBD, ``Backdoorbench: A comprehensive benchmark of backdoor learning,'' https://github.com/SCLBD/backdoorbench, 2022, gitHub repository, accessed 2026-05-30.