Testing the Limits: Exploring Adversarial Techniques in AI Models

Apostolis Zarras[1], **Athanasia Kollarou**[1], **Aristeidis Farao**[2], **Panagiotis Bountakas**[1], **and Christos Xenakis**[1]

[1]**University of Piraeus, Greece**
[2]**InQbit Innovations SRL, Romania**

Corresponding author:
Apostolis Zarras[1]

Email address: zarras@unipi.gr

ABSTRACT

The rising adoption of Artificial Intelligence and Machine Learning in critical sectors underscores the pressing need for robust systems capable of withstanding adversarial threats. While deep learning architectures have revolutionized tasks such as image recognition, their susceptibility to adversarial techniques remains an open challenge. This article evaluates the impact of various adversarial methods, including the Fast Gradient Sign Method, Projected Gradient Descent, DeepFool, and Carlini & Wagner, on five neural network models: a Fully Connected Neural Network, LeNet, Simple CNN, MobileNetV2, and VGG11. Using the EvAIsion tool explicitly developed for this research, these attacks were implemented and analyzed based on accuracy, F1 score, and misclassification rate. The results revealed varying levels of vulnerability across the tested models, with simpler architectures occasionally outperforming more complex ones. These findings emphasize the importance of selecting the most appropriate adversarial technique for a given architecture and customizing the associated attack parameters to achieve optimal results in each scenario.

1 INTRODUCTION

In recent years, Artificial Intelligence (AI) and Machine Learning (ML) have gained considerable popularity and have been integrated into a wide range of sectors, transforming the way many industries operate by powering applications (Bountakas et al., 2023) in areas such as image recognition, natural language processing, healthcare, and autonomous vehicles. These applications rely on algorithms that analyze large volumes of training data to identify patterns and make predictions. Deep Learning (DL), a subfield of ML, utilizes multilayered neural networks to handle complex data and solve intricate problems. This capability has proved highly effective in tasks including image classification and speech recognition (Charalambous et al., 2022).

Despite the remarkable advances in AI and ML, these models exhibit specific vulnerabilities that can compromise their reliability and robustness. The complexity of these systems poses significant challenges in ensuring smooth operation while upholding high-security standards, an issue of particular concern in domains where models support critical functions. In this context, models face significant limitations arising from their inherent operational characteristics and the intentional exploitation of these weaknesses through adversarial techniques (Petihakis et al., 2024).

From an internal perspective, the sensitivity of DL architectures to minimal perturbations can result in misclassifications or unpredictable behavior (Suciu et al., 2022). This challenge becomes even more pressing when models fail to generalize effectively to real-world conditions, such as noisy or anomalous data, often due to overfitting on curated datasets. Furthermore, the opacity of these models, often called the black-box problem, hampers interpretability and undermines trust in the decision-making process. Externally, adversarial techniques exploit these vulnerabilities by introducing stealthy modifications that cause models to misbehave at inference time (evasion attacks), corrupt training data (poisoning attacks), or steal proprietary models or sensitive information (model extraction and inference attacks) (Bountakas et al., 2023).

A particularly concerning category of attack involves adversarial examples. First described by Szegedy (Szegedy, 2013), these are intentionally crafted inputs containing undetectable perturbations to subvert the classification process. Subsequent research (Lin et al., 2017; Dong et al., 2018) has reinforced these findings, demonstrating the ease with which adversarial examples can compromise real-world systems.

AI-based systems have been linked to various cybersecurity incidents over the years. In 2025, OmniGPT, an AI chatbot service, reportedly experienced a data breach in which a hacker claimed to have accessed users' billing information and credentials (csoonline, 2025). Similarly, in 2024, Muah.AI, a platform for creating AI-generated virtual partners, was compromised, affecting 1.9 million users, and exposing data suggesting that some individuals were generating illicit content (Linklaters, 2024). Furthermore, a study by Palisade Research revealed that in strategy games such as chess and go, older AI models (e.g., OpenAI's GPT-4o and Anthropic's Claude Sonnet 3.5) required prompting from researchers to attempt cheating tactics. However, more recent models, including o1-preview and DeepSeek R1, pursued exploits without prompting, indicating that AI systems may develop deceptive or manipulative strategies independently of explicit human instruction (TIME, 2025).

In response, several defense mechanisms have emerged to enhance the resilience of AI models against such threats (Bountakas et al., 2023; Pantelakis et al., 2023). A common strategy is *Adversarial Training* (Goodfellow et al., 2014; Hussain et al., 2025), which augments the training dataset with adversarial examples, helping models recognize and resist malicious inputs. *Distillation* (Hinton et al., 2015; Papernot et al., 2016) represents another popular approach, wherein outputs from one model are used to train another, smoothing decision boundaries and diminishing susceptibility to gradient-based attacks. Additional methods, such as *Gradient Masking* (Lee et al., 2020; Zhang et al., 2025), *Feature Compression* (Bhagoji et al., 2018; Chuah et al., 2022), and *Noise Reduction* (Joshi et al., 2022), focus on obfuscating the critical details exploited by attackers or removing adversarial changes prior to model processing.

Nonetheless, existing defense mechanisms suffer from limitations that constrain their real-world efficacy. For instance, adversarial training typically targets specific attack types and thus is less effective against evolving threats. Defensive distillation or similar methods can reduce a model's accuracy on benign data, which is problematic for applications demanding both robustness and high precision. Moreover, computational inefficiency remains a concern, particularly in large-scale or real-time applications, where the significant resource demands of defensive methods can be impractical. These challenges are compounded by the fragmented manner in which many defenses are tested, focusing on narrow attack scenarios or specific model types. Such a restricted perspective impedes our understanding of how diverse models fare across different adversarial conditions.

Overall, the identified AI/ML cybersecurity-related gaps can be summarized as follows: (i) Adversaries can manipulate inputs to deceive AI/ML models; (ii) AI/ML models often process sensitive data, making them attractive cybersecurity targets; (iii) Adversaries can steal, replicate, or exploit proprietary AI models; (iv) Many AI models function as black boxes, complicating the detection of malicious activities; (v) Excessive reliance on third-party datasets, libraries, and hardware, which may have been previously compromised; (vi) AI systems are often based on minimal to no prior robustness tests (i.e., against adversarial attacks). In response to these gaps, this article investigates the impact of various adversarial techniques on different neural network architectures. Specifically, we have designed and implemented EvAIsion, a custom evaluation tool engineered to execute adversarial attacks systematically on selected architectures and evaluate their resilience using predefined performance metrics[1].

In summary, we make the following main contributions:

- We categorize existing adversarial techniques that aim to exploit vulnerabilities in ML and DL models.
- We define security requirements for tools dedicated to performing adversarial AI attacks to ensure the tools' proper functionality and ethical use.
- We design and develop EvAIsion, a tool for executing adversarial techniques and testing models.

[1]The source code of EvAIsion can be found at `https://github.com/UniPiSSL/testing-the-limits-evAIsion`

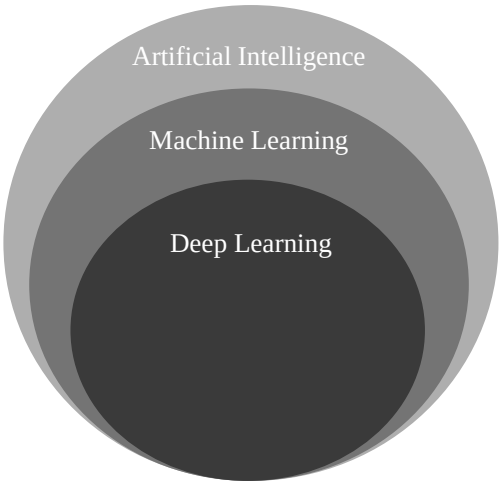

Figure 1. Correlation among AI, ML, and DL

- We perform a comparative analysis of adversarial AI evasion attacks in three discrete datasets (MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky, 2009)).

The remainder of this article is organized as follows. Section 2 outlines the key theoretical foundations and provides the contextual background necessary for understanding this research. Section 3 details the methodological approach. Section 4 presents the results and discusses the key findings. In Section 5, the real-world implications of this study are examined, while Section 6 discusses the limitations of the proposed approach and suggests avenues for future research. Section 7 reviews relevant literature. Finally, Section 8 concludes the article, summarizing the main contributions.

2 BACKGROUND

The rapid advancement of AI and ML has enabled their adoption across a broad spectrum of fields, including healthcare, finance, and autonomous systems. Despite these transformative capabilities, AI models remain susceptible to adversarial threats that can undermine their integrity and reliability. Consequently, a solid understanding of the foundational principles of AI, its core methodologies, and the associated security challenges is essential for evaluating the robustness of these models against adversarial attacks. This section provides an overview of fundamental AI and ML concepts, examining the key components of DL architectures. Following this, we introduce the concept of Adversarial Machine Learning (AML), which examines adversaries' techniques to exploit model vulnerabilities. By establishing this foundational knowledge, we highlight the importance of adversarial threats and their broader implications for AI-driven systems.

2.1 Fundamentals

AI encompasses the design of computational systems capable of performing tasks that typically require human intelligence, such as decision-making, problem-solving, and pattern recognition. At the core of AI lies ML (see Figure 1), a subset that trains models to learn patterns from data in order to make predictions or classifications. Within ML, DL constitutes a specialized subfield that leverages *Artificial Neural Networks* with multiple layers to capture complex relationships in data.

DL architectures such as Convolutional Neural Networks (CNNs) are particularly well-suited for tasks involving image data. These networks transform input data through a series of interconnected layers, relying on two key operations: (i) *convolution*, a mathematical operation that extracts spatial features, enabling the network to detect edges, textures, and other distinctive patterns (LeCun et al., 1998) and (ii) *pooling*, which reduces the spatial dimensions of feature maps while retaining essential information, thereby decreasing computational complexity (Krizhevsky et al., 2012).

To enable effective learning, activation functions, such as the Rectified Linear Unit (ReLU), introduce non-linearity by mapping negative inputs to zero, thereby allowing the model to capture more complex

patterns. Another central concept in DL is optimization, where the model's parameters are iteratively adjusted to minimize a loss function, such as cross-entropy loss. This loss function measures the discrepancy between the model's predictions and the actual labels, guiding training toward improved performance (Goodfellow et al., 2016).

Optimization algorithms like Stochastic Gradient Descent (SGD) and Adam help fine-tune the model's weights efficiently. While SGD updates parameters incrementally on small batches of data, Adam employs adaptive learning rates, making it particularly effective for large-scale datasets. By applying these techniques, CNNs have achieved exceptional results in image classification tasks. For instance, models trained on the MNIST dataset, which consists of grayscale images of handwritten digits, are widely used to benchmark classification accuracy and evaluate resilience to adversarial attacks (LeCun et al., 1998).

2.2 Adversarial Machine Learning

AML primarily investigates the vulnerabilities of ML models and their capacity to withstand intentionally crafted inputs, often referred to as adversarial examples (Bountakas et al., 2023; Farao et al., 2024). These examples involve subtle perturbations that prompt models to make erroneous predictions, even though the modifications are nearly imperceptible to the human eye. Adversarial examples can severely undermine ML systems in numerous domains (Wang et al., 2023); for instance, minimal alterations to an image can lead a CNN to misclassify the image with high confidence. Even a slight adjustment to an image of the digit `3` can cause the model to mistakenly recognize it as a `5`.

Such adversarial techniques can also be employed in physical settings. Examples include using laser beams to manipulate traffic sign recognition systems or crafting adversarial Channel State Information (CSI) inputs to mislead IoT-based Deep Neural Networks (DNNs). Adversarial attacks may target various stages of the ML lifecycle. During the training phase, poisoning attacks involve injecting malicious data into the training set, thereby producing compromised or biased models (Biggio et al., 2012). In the testing or inference phase, evasion attacks modify inputs in ways that force the model to misclassify, even though these modifications are often imperceptible to human observers (Goodfellow et al., 2016). Furthermore, adversarial methods such as model extraction and inference attacks directly exploit the model by replicating its behavior or extracting sensitive information (Tramèr et al., 2016).

To address these threats, researchers are actively exploring various defensive strategies to bolster model robustness. Wu et al. (Wu et al., 2023) systematically review these approaches, categorizing them based on their position in the ML lifecycle. Their framework highlights defenses that span pre-training, training, and post-training stages, emphasizing a proactive and holistic strategy for safeguarding ML systems.

2.3 Adversarial Techniques

Adversarial techniques comprise a broad spectrum of methods that exploit vulnerabilities in ML models to degrade their performance or extract sensitive information. These methods can manifest at any stage of the ML lifecycle, spanning from the contamination of training data to the manipulation of model inputs during inference. Moreover, adversarial methods are often classified by the attacker's level of knowledge about the model (yielding white-box and black-box attacks) or by the type of output being manipulated, such as scores or decisions. In this subsection, we examine the principal categories of adversarial techniques, including evasion attacks, poisoning attacks, model extraction attacks, and inference attacks, and underscore how each class exploits distinct vulnerabilities within ML systems.

2.3.1 Overview of Adversarial Techniques

Adversarial techniques exploit vulnerabilities in ML models to compromise their performance or extract sensitive information. These methods encompass various approaches to deceive or manipulate models at various stages of their lifecycle, from training to deployment. Adversarial techniques can be categorized using multiple criteria, including the adversary's knowledge of the model (i.e., knowledge-based categories), the type of model output, or the specific goals of the attack, among others.

Based on the adversary's knowledge of the model, adversarial techniques are commonly categorized into white-box attacks and black-box attacks (Kotyan, 2023). In white-box attacks, the adversary possesses comprehensive knowledge of the model, including its architecture and parameters. Such insight enables the precise crafting of adversarial examples to exploit the model's vulnerabilities. A well-known technique in this context is the Fast Gradient Sign Method (FGSM), which calculates the loss gradients with respect

to the input to produce malicious perturbations. Particularly, in the *white-box pipeline*, the adversary is assumed to possess complete knowledge of the target model, including its architecture, parameters, and the ability to compute exact input gradients via backpropagation. Within this setting, the adversarial objective can be formulated as either untargeted or targeted. In the untargeted case, the adversary seeks to maximize the classification loss in order to induce any misclassification, whereas in the targeted case, the objective is to minimize the loss toward a specific target class. Additionally, all adversarial examples are restricted to the valid input domain to preserve semantic similarity with the original inputs. Given clean samples and their corresponding ground-truth labels, the preprocessing steps applied by the model, such as normalization, resizing, or data type transformations, must be accurately replicated during the attack to ensure correct gradient computation. The perturbation is then obtained by optimizing the chosen objective function using gradient-based techniques. The success of generated adversarial inputs is typically evaluated using metrics such as the accuracy for untargeted attacks, or the percentage of inputs classified into the desired target class for targeted attacks. Overall, the white-box attack pipeline represents the most powerful adversarial setting, establishing an *upper bound* on model vulnerability and serving as a benchmark for evaluating the robustness of machine learning systems.

By contrast, black-box attacks occur when the adversary lacks direct access to the model's internal structure and parameters (Bountakas et al., 2023). Instead, the attacker must rely on querying the model and analyzing its outputs to infer decision boundaries. A common strategy involves training a surrogate model to approximate the target model's behavior, allowing adversaries to develop and evaluate adversarial examples without explicit knowledge of the original model's inner workings. More precisely, in the *black-box pipeline*, the adversary is assumed to have no access to the internal architecture, parameters, or gradients of the target model and can only interact with it through its outputs. Depending on the level of feedback available, black-box attacks can be categorized into three main strategies. The first is *transfer-based attacks*, where the adversary cannot directly query the target model but instead trains a surrogate model on data from a similar distribution. Adversarial examples are crafted on the surrogate model using white-box techniques are then transferred to the target model, relying on the transferability property of adversarial examples. The second category is *score-based attacks*, where the adversary can query the target model and obtain confidence scores or logits. In this setting, the attacker estimates approximate gradients by analyzing the variations in the model's output scores when small perturbations are applied to the inputs and then uses these estimated gradients to iteratively generate adversarial examples. The third category is *decision-based attacks*, where only the predicted class labels are accessible. These attacks typically start from a heavily perturbed input that is already misclassified and iteratively reduce the perturbation magnitude while ensuring the example remains adversarial, using techniques such as the Boundary Attack.

While white-box access is typically unattainable in real-world settings, attackers may still obtain limited information, such as a subset of input features, output class labels, or, in the case of DNNs, intermediate representations from hidden layers. This partial insight allows adversaries to develop more informed attack strategies than in black-box settings while operating under realistic constraints. This is widely known as the *gray-box attack*. Here, the adversary has partial knowledge of the target model but lacks full access to its internal parameters or complete architecture. In this setting, the attacker may know aspects such as the backbone network, data distribution, preprocessing techniques, or normalization statistics, but other components, such as task-specific heads or stochastic defenses, remain unknown. Limited queries to obtain labels or confidence scores may also be permitted under strict constraints. The attack can be either untargeted, aiming to induce any misclassification, or targeted, forcing predictions into a specific class, while ensuring perturbations remain visually imperceptible and valid within the input domain. To exploit available knowledge, the adversary typically builds a calibrated surrogate model aligned with the known properties of the target system. When possible, the surrogate is refined through fine-tuning, synthetic data generation, or distillation from limited queries. For non-differentiable components or randomized defenses. Compared to white-box attacks, gray-box scenarios are more challenging but also more realistic, bridging the gap between fully transparent systems and complete black-box settings.

Adversarial attacks can also be grouped by their strategy, defining their implementation and objectives. *Evasion attacks* introduce small, often imperceptible perturbations to data during inference to mislead the model. *Poisoning attacks* compromise the training process by injecting malicious samples into the dataset. *Model extraction attacks* replicate a proprietary model by querying it and subsequently training a

substitute. Finally, *inference attacks* (e.g., membership or model inference and attribute inference) aim to glean sensitive information from the model, such as whether a specific record was included in the training set.

2.3.2 Categories of Adversarial Techniques

Adversarial techniques exploit specific vulnerabilities in ML models, often targeting their behavior in various ways. While the spectrum of adversarial methods is extensive, this section concentrates on four key attack strategies—evasion attacks, poisoning attacks, model extraction attacks, and inference attacks—that exemplify adversaries' diverse approaches to compromise models.

Evasion Attacks. Evasion attacks exploit ML models' vulnerabilities during inference by introducing small, carefully designed perturbations to the input data. While these perturbations usually remain imperceptible to the human eye, they are crafted to deceive the model into making incorrect predictions or classifications. By targeting a model's decision boundaries, attackers can push inputs across these boundaries with minimal modifications, thereby causing a significant degradation in the model's performance. For instance, an adversary could subtly alter an image of a stop sign so that a computer vision model, potentially integrated into an autonomous vehicle, misclassifies it as a yield sign (Papernot et al., 2017). Such scenarios can lead to dangerous real-world consequences.

Evasion attacks fundamentally rely on gradient-based methods. Attackers compute the gradient of the model's loss function with respect to the input data to determine the direction that maximally increases the model's error. By applying a perturbation aligned with this direction, they create adversarial examples that appear visually unchanged to human observers. A commonly used formulation for crafting such perturbations is shown in Equation 1 (Pantelakis et al., 2023):

$$X_{adv} = x + \in \bullet sign(\bigtriangledown_x L(x,y)) \tag{1}$$

where x is the original input, $\in$ is the perturbation magnitude, which determines the quantity of noise that is added to the input, and $\bigtriangledown_x L(x,y)$ is the gradient of the loss function with respect to x.

The effectiveness of evasion attacks stems from deep learning models' inherent sensitivity to slight input variations. While this sensitivity facilitates the capture of complex patterns, it also makes these models vulnerable to adversarial manipulations. A thorough understanding of evasion attacks helps researchers anticipate potential threats and develop strategies to fortify model robustness.

Poisoning Attacks. Unlike evasion attacks, which target the model during inference, poisoning attacks involve intentionally manipulating training data to introduce vulnerabilities into the model. These attacks exploit the reliance of ML algorithms on clean and representative datasets by contaminating the training set with maliciously crafted samples. The adversary strategically injects these samples to influence the model's learning process by degrading its overall performance or inducing specific erroneous behaviors under targeted conditions.

Poisoning attacks typically exploit vulnerabilities in the data collection and model training pipeline by leveraging the following mechanisms:

- *Label Flipping:* Adversaries manipulate the labels of training samples to induce incorrect associations within the model. For example, altered labels in a facial recognition system may lead the model to misidentify individuals, thereby degrading its classification accuracy.
- *Feature Injection:* Attackers introduce malicious examples containing irrelevant or misleading features into the training dataset. Such perturbations distort the model's feature space and learning trajectory. For instance, inserting benign-looking but carefully crafted patterns into spam emails can cause the classifier to misinterpret or overlook legitimate spam indicators.
- *Backdoor Attacks:* These constitute a specialized class of poisoning attacks, wherein adversaries embed specific "triggers" within the data that are covertly associated with particular target labels. During inference, the trigger compels the model to misclassify inputs, irrespective of their true content.

Poisoning attacks pose a significant threat, mainly when training data is sourced from unverified or publicly accessible origins. Such vulnerabilities are prevalent in federated learning frameworks, open-source repositories, and collaboratively curated datasets. In these contexts, attackers can insert adversarial samples into the data stream, compromising the resulting model's integrity and reliability.

Model Extraction Attacks. These target ML systems by replicating their functionality through systematic querying. These attacks are particularly concerning for models deployed via publicly accessible APIs, where adversaries can exploit the query-response interface to infer decision boundaries or approximate the underlying parameters of the target model. Such actions not only compromise the intellectual property of the model owner but also enable further adversarial activities, such as evasion attacks.

The tactics employed by attackers are influenced by the type of model outputs available. When soft labels (i.e., class probabilities) are accessible, adversaries can efficiently approximate model parameters with relatively few queries. In contrast, when only hard labels (i.e., predicted classes) are returned, the extraction process becomes more challenging. Nonetheless, Tramèr et al. (Tramèr et al., 2016) demonstrated that linear and non-linear models can be effectively extracted even in such constrained settings. Their work showed that it is possible to replicate a target model's behavior with high fidelity despite limited access to output information with carefully crafted queries.

Model extraction is often accompanied by surrogate training, wherein the adversary uses the collected query-response pairs to train a local model that mimics the decision-making process of the target system. This surrogate model can be exploited for unauthorized deployment, competitive advantage, or as a stepping stone for further attacks.

Mitigation strategies against model extraction attacks typically involve reducing the amount of information disclosed through APIs. This may include returning only hard labels instead of probability distributions, applying rate limiting to restrict the volume of queries, and incorporating noise or differential privacy techniques to obscure outputs. These defenses balance system usability with robust protection against unauthorized model replication.

Inference Attacks. Finally, inference attacks target the privacy and confidentiality of ML systems by extracting sensitive information related to the training data, model parameters, or other private aspects. These attacks exploit the information revealed through model predictions and pose significant risks in sensitive domains such as healthcare, finance, and legal systems, where data confidentiality is paramount.

A prominent category of inference attacks is *membership inference*, wherein adversaries aim to determine whether a specific data point was included in the training set (Shokri et al., 2017). This is typically achieved by analyzing prediction outputs, such as confidence scores or probability distributions. Membership inference attacks are particularly concerning in contexts involving personal or medical data, as they may lead to severe privacy violations.

Another notable variant is *attribute inference*, where attackers attempt to deduce sensitive input data attributes, such as demographic information or behavioral characteristics, even when these attributes are not explicitly present in the dataset. Furthermore, some inference attacks extend to extracting model parameters, thereby exposing proprietary configurations and enabling unauthorized replication or exploitation.

Overfitting often facilitates these attacks, as models that memorize training data exhibit greater vulnerability to inference. Consequently, the susceptibility of ML models to inference attacks highlights the need for robust defense mechanisms. Techniques such as *differential privacy*, which introduces controlled noise to model outputs, can obscure sensitive information, while output restrictions, such as providing hard labels instead of confidence scores, can further mitigate information leakage. Additionally, regularization methods that reduce overfitting enhance model generalization and serve as a preventive measure against inference-based exploits.

2.3.3 Diving into Evasion Attacks

Evasion attacks are among the most prominent adversarial strategies that exploit the vulnerability of ML models to input perturbations at the inference stage. This category of attacks poses significant challenges, especially in critical applications where reliability is paramount. In this section of the works, four evasion attack techniques will be studied, including FGSM, Projected Gradient Descent, DeepFool, and Carlini & Wagner (C&W). Each of these techniques represents a unique approach to generating adversarial examples.

Fast Gradient Sign Method. The FGSM (Goodfellow et al., 2014) is one of the earliest and most extensively studied adversarial attack techniques. It exploits the sensitivity of DNNs to small, deliberately crafted perturbations in input data, resulting in misclassification, while the perturbations remain imperceptible to human observers. FGSM is classified as a white-box attack, assuming the attacker fully knows the model's architecture and parameters. The simplicity and computational efficiency of FGSM have made

it a popular choice for adversarial research and benchmarking model robustness. The core idea behind FGSM is to perturb the input data in the direction of the gradient of the loss function with respect to the input. This method modifies the input so as to maximize the model's prediction error, thereby inducing misclassification. The adversarial example x_{adv} is computed as detailed below (see Equation 2) (Pantelakis et al., 2023):

$$x_{\text{adv}} = x + \varepsilon \cdot \text{sign}(\nabla_x L(x, y)) \tag{2}$$

where x is the original input, ε is the perturbation magnitude controlling the amount of noise added to the input, $\nabla_x L(x, y)$ denotes the gradient of the loss function L with respect to the input x, and y is the true label associated with the input x.

The FGSM attack proceeds as follows: the original input x is first passed through the model to compute the loss $L(x, y)$ using the true label y. Next, the gradient $\nabla_x L(x, y)$ is calculated, indicating how changes in the input affect the loss. The sign of this gradient, $\text{sign}(\nabla_x L(x, y))$, is then used to determine the direction in which the loss increases most rapidly. ε scales this directional information to control the perturbation's magnitude. Finally, the adversarial example x_{adv} is generated by adding the scaled perturbation to the original input x.

Due to its effectiveness, speed, and ease of implementation, FGSM is a foundational technique in adversarial machine learning and is often employed as a baseline for evaluating the robustness of ML models.

Projected Gradient Descent. This is an iterative adversarial attack method designed to generate adversarial examples by optimizing perturbations within a constrained range. It is considered a more powerful and generalized extension of the FGSM. Projected Gradient Descent (PGD) operates by iteratively refining perturbations to maximize adversarial impact while ensuring that the modified input remains within a specified ε-bounded region around the original input. This makes PGD particularly effective at crafting adversarial examples that remain imperceptible while successfully misleading the model.

PGD assumes full access to the target model's architecture and parameters, and is therefore categorized as a white-box attack. The attack process involves two key steps: computing the gradient of the loss function with respect to the input and projecting the perturbed input back into the allowed ε-ball to ensure it remains within the specified constraint.

The attack begins by adding a small random perturbation δ within the ε-ball around the original input x, initializing $x_0 = x + \delta$. For each iteration t, the gradient of the loss function $L(x_t, y)$ is computed with respect to the current input x_t, where y is the true label. The input is then updated as follows (see Equation 3) (Madry et al., 2017):

$$x_{t+1} = \Pi_{\mathscr{B}_\varepsilon(x)}\left(x_t + \alpha \cdot \text{sign}(\nabla_x L(x_t, y))\right) \tag{3}$$

where α is the step size controlling the magnitude of each perturbation update, $\Pi_{\mathscr{B}_\varepsilon(x)}(\cdot)$ denotes the projection operator that ensures the updated input remains within the ε-ball $\mathscr{B}_\varepsilon(x)$ centered at the original input x, and $\nabla_x L(x_t, y)$ is the gradient of the loss function with respect to x_t.

After each gradient update, the projection step ensures the perturbed input stays within the allowable perturbation range. This iterative process continues for a predetermined number of steps or until the adversarial example causes the model to misclassify the input.

The key strength of PGD lies in its iterative nature, which allows for more precise perturbation refinement compared to single-step methods like FGSM. As a result, PGD can generate highly effective adversarial examples that are significantly more challenging to defend against. Due to its effectiveness and generality, PGD is often regarded as a benchmark for evaluating model robustness against adversarial attacks.

DeepFool. DeepFool is an iterative adversarial attack that seeks to find the minimal perturbation required to misclassify an input. Initially proposed by Moosavi-Dezfooli et al. (Moosavi-Dezfooli et al., 2016), the core idea involves locally linearizing the classifier's decision boundaries at each iteration and progressively perturbing the input towards the nearest decision boundary until the model's prediction changes. Unlike gradient-based methods such as FGSM and PGD, which apply fixed or bounded perturbations, DeepFool is designed to compute the smallest possible perturbation that causes misclassification. This often results in adversarial examples virtually imperceptible to human observers due to the minimal perturbation magnitude.

DeepFool was initially developed for binary classifiers, but its methodology can be extended to multi-class models. The attack proceeds iteratively by determining the smallest perturbation that shifts the input across the decision boundary. Let x be the original input, correctly classified with the true label y, and denote the initial input as $x_0 = x$. At each iteration t, the classifier's decision boundary is approximated by linearizing the output function $f(x)$ at the current point x_t. For linear models, this step precisely reveals the decision boundary; for non-linear models, it yields a local linear approximation of the decision surface.

At iteration t, the algorithm computes the perturbation r_t needed to move x_t across the closest linearized decision boundary. This perturbation is calculated using Equation 4 (Moosavi-Dezfooli et al., 2016):

$$r_t = \frac{|f(x_t)|}{|\nabla f(x_t)|^2} \cdot \nabla f(x_t) \tag{4}$$

where $f(x_t)$ represents the classifier's output at x_t, and $\nabla f(x_t)$ denotes the gradient of the output with respect to the input. Once r_t is computed, the input is updated as $x_{t+1} = x_t + r_t$, effectively moving it closer to the decision boundary than the previous input x_t. This process of linear approximation, perturbation calculation, and input update is repeated iteratively until the model's prediction for x_t changes. The total perturbation required to induce misclassification is then given by $r = x_t - x$, representing the minimal adversarial adjustment necessary.

Carlini and Wagner. The adversarial attack proposed by Carlini and Wagner (C&W) (Carlini and Wagner, 2017) is regarded as one of the most effective and extensively studied methods in adversarial machine learning. This attack is distinguished by its capacity to bypass a broad range of defense mechanisms while introducing minimal, often imperceptible perturbations. The C&W attack prioritizes the generation of adversarial examples that are both misclassified by the target model and minimally altered from the original input, with perturbations typically constrained under specific norm bounds.

In contrast to simpler gradient-based methods, such as the FGSM, the C&W attack employs a more rigorous optimization framework. The core of this approach lies in formulating an objective function that jointly minimizes the perturbation magnitude and maximizes the likelihood of misclassification. Perturbation size, denoted δ, is most commonly quantified using the L_2 norm, which measures the Euclidean distance between the original input and the perturbed sample. Alternatively, the L_∞ norm (which captures the largest absolute change across all pixels) or the L_0 norm (which counts the number of altered pixels, emphasizing sparsity) may be used, depending on the attack variant.

Formally, the adversarial example x_{adv} is computed by adding a perturbation vector r to the original input x, as shown in Equation 5 (Carlini and Wagner, 2017):

$$x_{\text{adv}} = x + r \tag{5}$$

where the perturbation r is constrained to ensure that its norm does not exceed a specified threshold, thereby maintaining imperceptibility. The misclassification constraint is enforced through a tailored loss function that encourages x_{adv} to be confidently assigned to a target class (for targeted attacks) or simply misclassified (for untargeted attacks). This objective function is minimized using gradient-based optimization techniques such as Adam or SGD.

A key aspect of the C&W attack is the incorporation of input validity constraints. For instance, in image data, pixel values must lie within a valid range, typically $[0, 1]$. To satisfy this, the attack introduces a change of variables, optimizing over an unconstrained parameter w instead of directly optimizing r. The adversarial example is then obtained via a transformation involving the hyperbolic tangent function, ensuring the resulting values stay within permissible bounds, as shown in Equation 6 (Carlini and Wagner, 2017):

$$x_{\text{adv}} = \frac{1}{2} \cdot (\tanh(w) + 1) \tag{6}$$

The optimization proceeds iteratively until convergence, at which point x_{adv} represents a minimally perturbed input that causes the model to err. The C&W attack's precision and adaptability have made it a benchmark for evaluating the robustness of ML models against adversarial threats.

3 DESIGN AND DEVELOPMENT

We design and develop a tool, named EVAISION, to enable the seamless execution and evaluation of four evasion attacks across various ML models. The current implementation supports the FGSM, PGD, DeepFool, and C&W attacks, applied to five distinct ML models, as detailed below. EVAISION facilitates a comprehensive evaluation of the effectiveness of these attacks using a range of performance metrics. The development process leverages the Adversarial Robustness Toolbox (ART) (IBM, 2025) and PyTorch (The Linux Foundation, 2025), which provide foundational functionalities for adversarial attack implementation.

3.1 Description of Supported Models

EVAISION is implemented with a modular architecture, enabling the seamless integration and evaluation of various models. Three benchmark datasets were employed for training and evaluation: MNIST[2], Fashion-MNIST[3], and CIFAR-10[4]. The MNIST dataset comprises grayscale images of handwritten digits, normalized to the range [0, 1]. Fashion-MNIST similarly contains grayscale images, depicting various clothing items. In contrast, CIFAR-10 consists of color images with three RGB channels.

Before training, each dataset undergoes a series of preprocessing steps to ensure model compatibility. These steps include normalization, resizing input images to match model-specific input dimensions, and data augmentation techniques, such as horizontal flipping, rotation, and cropping, to enhance model generalization and reduce overfitting. Then, model training is conducted using a batch size of 64. Optimizers are selected based on the model architecture and complexity, with commonly used options including Adam and SGD.

3.1.1 Fully Connected Neural Network

The Fully Connected Neural Network (FCNN) serves as a baseline model within the EVAISION framework. It comprises a sequence of layers that operate on flattened input vectors, with a straightforward architecture that renders it well-suited for both grayscale datasets, such as MNIST and Fashion-MNIST, and RGB datasets, such as CIFAR-10, following appropriate preprocessing. Specifically, this model employs the ReLU activation function in its hidden layers and utilizes the cross-entropy loss function during training.

In the FCNN, each input image is first transformed into a one-dimensional vector before propagating through three fully connected layers. ReLU activations are applied between layers to introduce non-linearity and facilitate learning complex representations. Due to the network's requirement for flattened inputs, datasets undergo preprocessing as follows: (i) MNIST and Fashion-MNIST images are directly reshaped from their original 2D format into 1D vectors and (ii) CIFAR-10 images, originally in RGB format, are first converted to grayscale to conform to the model's expected input structure, after which they are flattened.

The architecture of the FCNN used in this work is composed of the following layers:

1. The first fully connected layer (*FC1*) maps the flattened input vector to 128 neurons.
2. The second layer (*FC2*) further reduces the dimensionality by mapping to 64 neurons.
3. The final output layer (*FC3*) produces 10 outputs, each corresponding to one of the 10 classes present in the MNIST dataset.

3.1.2 LeNet

LeNet is a classical CNN architecture supported by EVAISION for assessing adversarial attacks. It comprises a series of convolutional layers followed by max-pooling operations, which progressively reduce spatial dimensions, culminating in fully connected layers for classification. The model employs the ReLU activation function and is optimized using the Adam optimizer to facilitate efficient and stable convergence during training.

Initially designed for grayscale image datasets such as MNIST and Fashion-MNIST, LeNet has been adapted in this work to support the CIFAR-10 dataset through appropriate preprocessing steps prior to input into the network.

[2] https://git-disl.github.io/GTDLBench/datasets/mnist_datasets/
[3] https://github.com/zalandoresearch/fashion-mnist
[4] https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

The LeNet architecture used in this study comprises *two convolutional layers* and *three fully connected layers*. Specifically, for the convolutional layers: (i) *Conv1* – Applies 6 filters of size 5×5 with padding to maintain the original spatial dimensions and (ii) *Conv2* – Applies 16 filters of size 5×5 to capture more complex features. Next, for the fully connected layers: (i) *FC1* – Transforms the flattened output from the convolutional layers into a 120-dimensional feature vector, (ii) *FC2* – Reduces the dimensionality to 84 neurons, and (iii) *FC3* – Outputs 10 neurons, corresponding to the number of target classes in MNIST.

The convolutional layers extract both low-level and high-level features from the input images. Max-pooling is applied after each convolutional layer to downsample feature maps and reduce computational complexity. The fully connected layers integrate these features and perform the final classification.

3.1.3 MobileNetV2

MobileNetV2 utilizes two convolutional layers to extract low- and high-level features from the images, while a maximum grouping layer helps reduce spatial dimensions after the second convolution. Its fully connected layers process the extracted features for final classification. In this article, MobileNetV2 is loaded using the Torchvision library (PyTorch Foundation, 2025).

Given that MobileNetV2 is designed to process three-channel images, preprocessing steps were applied to ensure compatibility across all datasets. *Input Adjustment:* Grayscale images from MNIST and Fashion MNIST were replicated across three channels to conform to the expected input format. *Output Adjustment:* The original classification head was replaced with a custom fully connected layer configured to output predictions for 10 classes, corresponding to digits (MNIST), clothing categories (Fashion-MNIST), or object classes (CIFAR-10).

3.1.4 VGG11

VGG11 is a well-established CNN architecture characterized by a sequential arrangement of convolutional layers followed by fully connected layers. Its structured and deep design has proven effective in tasks requiring robustness analysis, such as adversarial evaluation. In this work, VGG11 was also loaded from the Torchvision library.

To adapt VGG11 for use with all datasets in this study, the following modifications were implemented. *Input Layer Adjustment:* The first convolutional layer was modified to accept single-channel grayscale images for the MNIST and Fashion MNIST datasets. *Input Data Adjustment:* All images were resized to 224×224 pixels to match VGG11's expected input dimensions. *Output Adjustment:* The final fully connected layer was replaced with a new layer configured to output predictions for 10 classes, corresponding to the labels in MNIST, Fashion-MNIST, and CIFAR-10.

3.2 Security requirements

Defining security requirements for an adversarial AI evasion attack tool is essential to mitigate misuse, uphold ethical research standards, and maintain control over the tool's operation. These requirements facilitate controlled access, restrict excessive resource consumption, and ensure the reproducibility of attacks. Furthermore, they safeguard sensitive data, preserve the integrity of targeted models, and support forensic analysis to monitor and address potential risks. Based on recent literature (Pantelakis et al., 2023; Petihakis et al., 2024), the following security requirements have been identified:

S1 – Secure Model Interaction. The tool must utilize controlled APIs for interaction with target models, thereby preventing direct filesystem modifications. All inputs and outputs associated with model queries must be logged and verified through checksums to detect unauthorized alterations.

S2 – Execution Integrity. Modifications to the attack code or dynamic tampering of attack algorithms during execution must be strictly prohibited to ensure the integrity of the tool's operation.

S3 – Controlled Access. Prior to execution, the integrity of attack scripts must be verified through cryptographic signing mechanisms to prevent unauthorized code execution.

S4 – Resource and Abuse Controls. The tool must implement resource usage constraints to mitigate the risk of Denial-of-Service (DoS) attacks and must be capable of detecting and halting recursive attack chaining that could result in uncontrolled adversarial retraining.

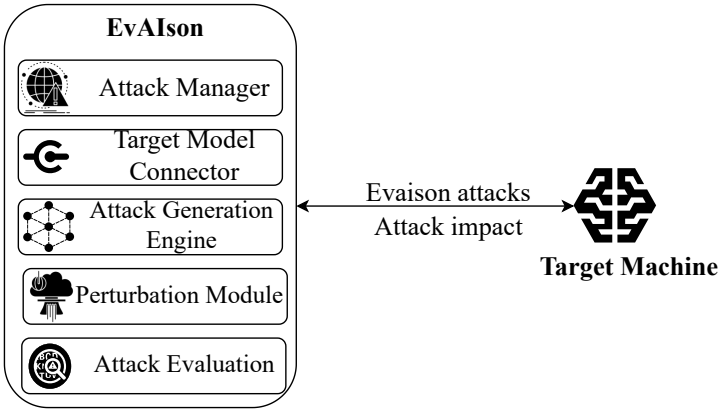

Figure 2. EVAISION blueprint

3.3 EVAISION Architecture

EVAISION is designed to perform adversarial attacks on ML models through an automated and modular framework. This design allows users to conduct multiple evaluations and collect performance metrics without developing separate scripts for each attack scenario. Its modular architecture ensures seamless integration of additional attacks, models, or evaluation metrics, making EVAISION inherently extensible for adversarial robustness testing. Figure 2 illustrates the architecture of EVAISION, highlighting its key components and their interactions.

The core components of EVAISION include the main script as well as the attack and evaluation modules.

Attack Manager. The Attack Manager acts as the central coordinator within EVAISION, overseeing the execution of adversarial attacks. It interfaces between user-defined parameters and internal components, ensuring a seamless workflow from input handling to attack execution and performance evaluation. A critical responsibility of the Attack Manager is to interface with the Target Model Connector, facilitating communication with external AI models. It manages the delivery of perturbed inputs and collects corresponding predictions for analysis. Moreover, the Attack Manager selects appropriate attack strategies based on user-specified requirements and model access constraints (e.g., white-box or black-box). It ensures the Attack Generation Engine operates using the selected methods, such as gradient-based approaches (e.g., FGSM, PGD). It also monitors the entire attack lifecycle, from input preprocessing and attack generation to results aggregation. The Attack Manager works closely with the Evaluation Module to guarantee accurate and meaningful performance insights, including attack success rates and perturbation metrics. Its adaptable design allows for straightforward expansion, accommodating diverse AI models and attack configurations, thereby providing a robust framework for evaluating model robustness.

Target Model Connector. This module serves as a standardized interface between EVAISION and external AI models, supporting both transparent (white-box) and opaque (black-box) settings. In transparent configurations, the connector enables direct access to gradients, model parameters, and architecture, facilitating highly optimized attacks. It supports query-based interactions in opaque scenarios, allowing EVAISION to send crafted inputs and analyze the model's outputs to identify vulnerabilities. The Target Model Connector also ensures input/output compatibility by applying necessary transformations, such as normalization, encoding, or resizing, tailored to the requirements of the target model. Overall, this module ensures reliable integration with diverse AI systems, enabling realistic adversarial testing across various deployment environments.

Attack Generation Engine. This module generates adversarial perturbations designed to deceive external AI systems. It implements diverse attack algorithms, categorized into gradient-based, optimization-based, and query-based techniques. It leverages methods such as FGSM, PGD, C&W, and DeepFool to exploit model vulnerabilities via carefully crafted input alterations in white-box scenarios. Its flexible design allows EVAISION to assess model robustness under various attack methodologies and operational conditions.

Data Perturbation Module. This module ensures that perturbations remain subtle yet effective in misleading the target model. It preprocesses input data to meet the format requirements of both the attack algorithms and the target AI system, applying transformations such as normalization, resizing, and encoding based on the data type (e.g., image, text, or structured data). Once preprocessed, the module applies adversarial perturbations generated by the Attack Generation Engine, adhering to specific norm constraints to maintain imperceptibility. It incorporates refinement techniques, including optimization-based adjustments, to balance minimal distortion with high attack success rates. This module ensures that generated adversarial examples are both realistic and impactful.

Attack Evaluation Module. This component evaluates the efficacy of adversarial attacks by analyzing the impact of perturbed inputs on the target model's predictions. It provides quantitative and qualitative metrics, including accuracy, precision, recall, F1 score, misclassification rate, and mean confidence. Additionally, it measures shifts in prediction confidence, highlighting how adversarial inputs affect model certainty. The module supports side-by-side comparisons between clean and adversarial predictions, enabling detailed analysis of model vulnerabilities. This module strengthens the analysis of adversarial robustness and supports broader security assessments by providing a comprehensive evaluation framework.

Processing Flow. The tool's processing pipeline begins with user-defined inputs and attack parameters, such as attack type (e.g., FGSM, PGD, C&W), perturbation constraints, and access mode (white-box or black-box). The input data is first processed by the Data Perturbation Module, which performs necessary preprocessing (e.g., normalization, resizing, encoding, tokenization). The preprocessed data is passed to the Attack Generation Engine, which generates adversarial perturbations in accordance with the selected strategy. In white-box settings, the engine computes gradients to optimize perturbations; in black-box scenarios, it uses query-based techniques to refine inputs iteratively. The generated adversarial examples are then passed to the Target Model Connector, which interfaces with the external AI model to obtain predictions. The Attack Evaluation Module analyzes these outputs to compute the final metrics. Each attack type requires specific parameters, summarized as follows:

- *FGSM:* epsilon
- *PGD:* epsilon, step size, number of iterations
- *DeepFool:* maximum iterations, overshoot
- *C&W:* confidence, learning rate, maximum iterations

Upon completion of the attack, the adversarial examples are used to evaluate the model's robustness through the defined metrics.

FGSM Implementation. The `run_fgsm` function leverages the Adversarial Robustness Toolbox (ART) (IBM, 2025) to implement FGSM. The input is converted to a NumPy array as ART requires, and the FGSM class is instantiated with the classifier and epsilon (perturbation strength). The `fgsm.generate()` method applies perturbations that maximize the model's loss.

PGD Implementation. The `run_pgd` function similarly uses ART for PGD. After converting input data to NumPy arrays, the PGD class is configured with parameters such as maximum perturbation, step size, and iterations. Adversarial examples are generated iteratively and returned as PyTorch tensors.

DeepFool Implementation. The `run_deepfool` function implements DeepFool via ART. It initializes the DeepFool class with the target classifier and generates minimal perturbations required for misclassification. The outputs are returned to the main pipeline for evaluation.

C&W Implementation. The `run_carlini_wagner` function uses ART's `CarliniL2Method` class. Inputs are transformed into NumPy arrays, and the class is configured with confidence and iteration parameters. The generate method applies the C&W optimization algorithm, and results are returned for further analysis.

4 PERFORMANCE EVALUATION

The execution of all the attacks and the calculation of the metrics presented below were carried out on a machine equipped with a 13th Gen Intel Core i7 processor, an NVIDIA GeForce RTX 4060 graphics card, and 16 GB of RAM, running the Windows 11 Operating System.

In EVAISION, a set of evaluation metrics was defined to assess the performance of the ML models against the adversarial techniques employed. These metrics were computed for the model performance on clean datasets and on the adversarial examples generated by the attacks. This dual approach ensures a more comprehensive comparison of the results. The metrics currently included in the evaluation module of EVAISION are described below. To ensure robustness and mitigate stochastic variability, each adversarial AI attack was independently executed five times per model-attack pair. The reported results represent the average performance across these runs, and standard deviations are also provided where applicable to reflect the statistical consistency of the outcomes. This repetition enables a more reliable comparison between models and attacks, contributing to the statistical validity of the evaluation.

Accuracy (see Equation 7) measures the proportion of correctly classified samples out of the total dataset. It also functions as a general indicator of the model's predictive capability.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}} \tag{7}$$

Precision evaluates the proportion of true positive predictions out of all positive predictions made by the model (see Equation 8). It reflects the model's ability to avoid false positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{8}$$

Recall measures the proportion of true positive predictions out of the actual positives in the dataset (see Equation 9). It assesses the model's ability to identify all relevant instances without missing any.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{9}$$

F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both false positives and false negatives (see Equation 10).

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

Misclassification Rate complements accuracy by measuring the proportion of incorrect predictions (see Equation 11). It is defined as:

$$\text{Misclassification Rate} = 1 - \text{Accuracy} \tag{11}$$

Mean Confidence measures the average confidence level of the model in its predictions for the true class labels (see Equation 12). If the output probabilities are not normalized, a softmax function is applied to ensure the values represent proper confidence scores.

$$\text{Mean Confidence} = \frac{1}{N} \sum_{i=1}^{N} P(\text{True Class}_{x_i}) \tag{12}$$

where $P(\text{True Class}_{x_i})$ is the predicted probability for the true class of sample x_i.

Below, we present the overall impact summary per model and per dataset (MNIST, Fashion-MNIST, CIFAR-10). The impact is calculated as the difference between the metric value after the attack and the metric value before the attack (see Equation 13).

$$\text{Impact} = \text{Metric}_{\text{After}} - \text{Metric}_{\text{Before}} \tag{13}$$

Table 1. Baseline metrics (Initial performance on AI models)

Model	Value (%)								
	Accuracy			***F1 Score***			***Precision***		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	92	87	39	91	85	35	91	87	37
LeNet	91	82	48	93	81	45	94	82	48
Simple CNN	96	89	61	97	88	59	97	88	62
MobileNetV2	96	84	18	96	83	13	65	85	15
VG11	96	87	50	96	87	48	96	88	49
	Recall			***Mean Confidence***			***Misclassification Rate***		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	93	85	37	91	83	35	7	12	60
LeNet	94	82	45	92	77	36	6	17	51
Simple CNN	98	90	61	96	88	60	3	10	38
MobileNetV2	96	84	17	92	80	16	3	14	81
VG11	97	87	51	97	86	46	3	12	50

For our experiments, we carefully selected the hyperparameters of the employed attacks based on commonly adopted practices in the literature to ensure fair comparisons. More precisely, for the FGSM, the perturbation magnitude ε was set to 0.2, which controls the maximum allowed distortion added to the input while generating adversarial examples. Regarding PGD, we used $\varepsilon = 0.1$, a step size per iteration of 0.001, and the maximum number of optimization iterations was fixed at 40. In C&W, we set the confidence parameter to 0.1 to enforce a small separation margin from the decision boundary, and the maximum number of iterations was fixed at 10. Finally, for DeepFool, the maximum number of iterations was set to 100, with an ε value of *1e-6* to control the overshoot parameter. Additionally, the number of class gradients computed per iteration was set to 10 to efficiently approximate decision boundaries. These hyperparameters were chosen to balance attack strength and computational efficiency, ensuring a fair evaluation across different adversarial techniques.

4.1 Baseline

The initial phase of our evaluation involved measuring the performance of the models under their original, unperturbed conditions before introducing adversarial AI attacks. The assessment is based on the aforementioned evaluation metrics. Table 1 summarizes the baseline performance of the models (i.e., FCNN, LeNet, Simple CNN, MobileNetV2, and VG11) during their *correct operation*. Notably, these baseline results were obtained from experiments conducted five times each in the described environment. Simple CNN consistently achieved the highest accuracy across all datasets (i.e., MNIST, Fashion-MNIST, and CIFAR-10). In contrast, LeNet performed the worst on MNIST and Fashion-MNIST, while MobileNetV2 exhibited the lowest accuracy on CIFAR-10. Although the evaluation is conducted on widely-used benchmark datasets (i.e., MNIST, Fashion-MNIST, and CIFAR-10), these datasets are standard in adversarial ML research and provide a controlled, reproducible environment for rigorous comparison across attack techniques and models. Their established role in the literature ensures baseline validity while enabling future extension to more complex, real-world datasets. This pattern is further corroborated by other metrics, including the F1 score, precision, recall, mean confidence, and misclassification rate.

4.2 FGSM Impact Summary

The results of the FGSM attack across all evaluated models demonstrate considerable variability in the degree of impact on performance metrics, thereby highlighting the relative robustness and vulnerabilities inherent to different neural network architectures under adversarial conditions (see Table 2).

A comparative analysis of model performance under FGSM perturbations across all datasets reveals that susceptibility to adversarial attacks is highly architecture-dependent. Notably, the Simple CNN exhibited substantial vulnerability in most datasets, though the FCNN experienced even greater degradation in performance, particularly on the MNIST dataset. The FCNN's lack of convolutional layers, and consequently spatial awareness, forces it to process input as a flat vector, severely limiting its capacity to capture local patterns and increasing its sensitivity to FCNN's gradient-based perturbations.

In contrast, while Simple CNN incorporates convolutional layers that provide a degree of spatial awareness, its shallow architecture limits its capacity for robust feature extraction and the formation of

firm decision boundaries. This inadequacy renders it similarly vulnerable to adversarial noise, albeit marginally more robust than FCNN.

LeNet, among the simpler architectures, demonstrated the highest resilience. Its use of convolutional layers and max pooling not only facilitates spatial feature extraction but also contributes to noise reduction, improving its defense against FGSM perturbations. MobileNetV2 outperformed LeNet in adversarial robustness, attributable to its more advanced architectural elements, such as depthwise separable convolutions and residual connections, which enhance feature extraction efficiency and offer improved resistance to adversarial inputs.

Among all models tested, VGG11 exhibited the greatest resilience to FGSM attacks. Its deep architecture allows for hierarchical feature extraction, wherein early layers capture low-level features and deeper layers abstract complex patterns. This depth and its ability to form robust decision boundaries provide superior defense against adversarial perturbations compared to the other architectures evaluated.

4.3 PGD Impact Summary

The PGD attack is another adversarial technique evaluated in this study (refer Table 3). The PGD attack iteratively refines perturbations to maximize their impact on model predictions. The following analysis summarizes the performance of each model under the PGD attack.

The results of the PGD attack across all models highlight the varying levels of vulnerability, as well as the different impacts on evaluation metrics, as presented in Table 3. The Simple CNN model was the most affected by PGD, similar to the effect of the FGSM attack. Although the model incorporates convolutional layers, which provide spatial awareness, its shallow architecture limits its ability to construct robust decision boundaries. Similarly, the FCNN exhibited significant vulnerability due to its lack of spatial awareness, processing inputs as flat vectors and relying on global gradients for predictions. While FGSM caused a notable degradation in FCNN's performance, the PGD attack, with its iterative and more targeted perturbations, led to slightly improved performance compared to FGSM.

Surprisingly, VGG11 did not perform as well under PGD as anticipated, with considerable drops in evaluation metrics. The deeper architecture, which was expected to offer resilience, appeared vulnerable to PGD's iterative perturbations, potentially due to its reliance on learned patterns that may have become fragile under adversarial manipulation. In contrast, MobileNetV2 demonstrated strong resistance to PGD, outperforming both the simpler models (FCNN and Simple CNN) and even VGG11. This is likely attributable to its use of separable convolutions and residual connections, which enhance its ability to extract features efficiently.

LeNet performed the best under the PGD attack, exhibiting the smallest performance drops across all metrics. Its convolutional layers and max pooling mechanisms provide substantial spatial awareness and noise reduction. While lacking the depth of VGG11, LeNet's relatively simple architecture may have contributed to its robustness, as it did not overfit to specific patterns during training.

Table 2. Impact of the FGSM attack on the model performance metrics

Model	**Impact (%)**								
	Accuracy			***F1 Score***			***Precision***		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-68	-54	-24	-66	-53	-23	-62	-48	-22
LeNet	-11	-43	-29	-11	-40	-26	-12	-39	-32
Simple CNN	-84	-64	-43	-83	-62	-43	-83	-60	-43
MobileNetV2	-8	-65	-3	-9	-68	-2	-8	-67	-5
VG11	-4	-60	-31	-5	-58	-33	-3	-52	-34
	Recall			***Mean Confidence***			***Misclassification Rate***		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-67	-53	-25	-67	-50	24	68	54	24
LeNet	-10	-41	-27	-13	-39	29	11	43	29
Simple CNN	-83	-62	-45	-81	-62	43	84	64	43
MobileNetV2	-9	-69	-3	-9	-62	3	8	65	3
VG11	-5	-60	-34	-5	-58	31	4	60	31

Table 3. Impact of the PGD Attack on the Model Performance Metrics

Model	Impact (%)								
	Accuracy			*F1 Score*			*Precision*		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-33	-24	-32	-34	-26	-30	-31	-25	-31
LeNet	-1	-25	-37	-1	-26	-35	0	-21	-37
Simple CNN	-86	-45	-43	-85	-44	-45	-85	-42	-46
MobileNetV2	-3	-75	-11	-6	-74	-8	-5	-74	-17
VG11	-15	-74	-38	-15	-75	-37	-13	-73	-37
	Recall			*Mean Confidence*			*Misclassification Rate*		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-32	-25	-33	-32	-23	-23	33	24	32
LeNet	-1	-28	-37	-3	-21	-22	1	25	37
Simple CNN	-86	-46	-46	-84	-44	-40	86	45	43
MobileNetV2	-5	-73	-8	-3	-72	-8	3	75	11
VG11	-13	-75	-37	-16	-71	-33	15	74	38

Table 4. Impact of the DeepFool Attack on the Model Performance Metrics

Model	Impact (%)								
	Accuracy			*F1 Score*			*Precision*		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-91	-76	-27	-90	-73	-26	-90	-70	-27
LeNet	-93	-72	-27	-94	-74	-25	-94	-75	-25
Simple CNN	-72	-83	-44	-70	-83	-43	-61	-85	-44
MobileNetV2	-96	-54	-11	-95	-52	-12	-94	-49	-11
VG11	-93	-78	-31	-93	-78	-30	-92	-79	-30
	Recall			*Mean Confidence*			*Misclassification Rate*		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-91	-74	-29	-53	-46	-11	91	76	27
LeNet	-94	-74	-27	-52	-35	-10	93	72	27
Simple CNN	-72	-82	-45	-50	-42	-22	72	83	44
MobileNetV2	-96	-54	-17	-54	-37	-3	96	54	11
VG11	-94	-77	-33	-49	-41	-14	93	78	31

4.4 DeepFool Impact Summary

The results from the DeepFool attack indicate that all the models tested were significantly affected, exhibiting substantial degradation across all performance metrics (refer to Table 4).

The Simple CNN demonstrated the most negligible impact on most metrics among the models evaluated, outperforming the other architectures tested in this attack. Its relatively simple decision boundaries limited the ability of the DeepFool-generated adversarial samples to degrade performance, in contrast to the more complex models. Despite this relative resilience, the Simple CNN still experienced noticeable reductions in all performance metrics, as illustrated in Table 4.

The FCNN, while showing a considerable impact from the attack, exhibited performance comparable to, and in some cases slightly better than, deeper models such as VGG11 in specific metrics. Its simpler architecture and reliance on global gradients allowed it to retain a slight advantage in terms of stability under the DeepFool attack compared to more complex models.

In contrast, LeNet and VGG11 demonstrated similarly compromised performance under the DeepFool attack. Although VGG11's deeper architecture and more complex decision boundaries had previously provided it with some robustness, these features were insufficient in mitigating the effects of DeepFool's iterative adjustments. The attack exploited the model's complexity, resulting in significant misclassifications. LeNet, a shallower architecture that had shown resilience against earlier attacks, also failed to maintain its robustness under DeepFool's iterative precision.

Lastly, MobileNetV2 experienced the most significant degradation across all datasets, particularly in key metrics such as accuracy and mean confidence. Despite the advantages of residual connections in improving stability under simpler attacks like FGSM or PGD, these mechanisms did not provide the same level of protection against the DeepFool attack. Consequently, MobileNetV2's performance was substantially impaired.

Table 5. Impact of the C&W Attack on the Model Performance Metrics

Model	Impact (%)								
	Accuracy			*F1 Score*			*Precision*		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-11	-32	-12	-12	-32	-9	-11	-31	-6
LeNet	-6	-28	-18	-8	-28	-17	-7	-27	-17
Simple CNN	-86	-78	-45	-87	-78	-44	-85	-77	-46
MobileNetV2	-17	-75	-5	-19	-74	-2	-16	-73	-3
VG11	-2	-13	-15	-2	-15	-16	-1	-12	-14
	Recall			*Mean Confidence*			*Misclassification Rate*		
	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10	MNIST	F-MNIST	CIFAR-10
FCNN	-11	-32	-11	-5	-16	-5	11	32	12
LeNet	-8	-28	-19	-3	-13	-5	6	28	18
Simple CNN	-87	-80	-46	-60	-51	-25	86	78	45
MobileNetV2	-18	-74	-5	-10	-46	-1	17	75	5
VG11	-2	-15	-16	-1	-9	-8	2	13	15

4.5 C&W Impact Summary

The results of the C&W attack highlight varying levels of impact across the tested models, demonstrating their relative robustness to adversarial perturbations (see Table 5).

The C&W attack (see Table 5) induced varying degrees of disruption across the models, revealing their resilience to these adversarial perturbations. As expected, the Simple CNN model exhibited the greatest susceptibility, experiencing substantial performance drops across most metrics. While its convolutional layers provided some spatial awareness, its shallow depth and weak decision boundaries rendered it particularly vulnerable to this attack. LeNet, with its combination of convolutional layers and max pooling, displayed moderate robustness compared to other models; however, the targeted nature of the C&W attack ultimately led to misclassifications, as its simple decision boundaries were insufficient to resist the perturbations. MobileNetV2 also suffered a noticeable performance degradation under the C&W attack, positioning it below the FCNN in terms of robustness. Despite its separable convolutions and residual connections, which enhance feature extraction, MobileNetV2's architecture was not optimized for handling highly targeted perturbations like those induced by the C&W attack. Interestingly, the FCNN performed better than some more advanced models, including MobileNetV2. Its simple architecture and linear decision boundaries, typically seen as disadvantages, may have made it less sensitive to the precise nature of C&W perturbations. In contrast, VGG11 demonstrated the highest resilience to the C&W attack. Its deep architecture and hierarchical feature extraction effectively resisted targeted perturbations.

In comparison, the performance of various attacks showed distinct patterns. DeepFool caused the most consistent and severe performance drops, whereas PGD was effective but less reliable, particularly impacting simpler models. FGSM had a similar effect on simpler architectures but had minimal impact on deeper models. The C&W attack exhibited the least consistent impact, significantly affecting simpler models but leaving more complex models largely unaffected.

5 REAL-WORLD IMPACT OF ADVERSARIAL AI ATTACKS

In this section, we analyze the application of AI in critical infrastructures, such as healthcare, transportation, and security systems, highlighting the potential impact of adversarial AI attacks within these environments.

Self-driving vehicle systems that rely on AI/ML employ a combination of computer vision, sensor integration, and DL techniques to accurately perceive their surroundings and make timely driving decisions. These systems typically use CNNs to recognize road signs (e.g., STOP signs), detect pedestrians, and identify lane markings, while radar sensors help measure distances and detect objects (e.g., parked vehicles) in three-dimensional space. Through extensive training on large datasets, AI models learn to associate specific visual and sensory inputs with appropriate driving behaviors, such as stopping at red lights or maintaining a safe distance from other vehicles (Fang et al., 2003). However, adversarial AI attacks can exploit vulnerabilities in these models by introducing subtle, carefully crafted modifications to road signs, camera inputs, or sensor readings, which may lead to misinterpretations and result in unsafe driving behavior (The Register, 2025). For example, a stop sign that has been subtly altered may appear

visually identical to an observer, causing it to be misinterpreted as a speed limit sign. This could result in the vehicle failing to stop at an intersection, potentially leading to severe traffic accidents, including loss of life (Chowdhury et al., 2020; Cui et al., 2019).

Healthcare systems that leverage AI rely on advanced deep learning models to analyze medical images, such as X-rays, MRIs, and CT scans, to identify and diagnose various diseases (Algarni and Thayananthan, 2025; Shaheen, 2021). These models are trained on large datasets of labeled medical images, enabling them to recognize patterns associated with conditions like cancer, pneumonia, and fractures. By accurately identifying abnormalities, AI assists radiologists in making faster and more reliable diagnoses, thereby minimizing errors and improving patient outcomes. However, evasion-based adversarial attacks, including FGSM, PGD, C&W, and DeepFool, pose significant risks by subtly altering medical images, leading to incorrect diagnoses (Mahimai et al., 2025; Sharma and Kaushik, 2025). For instance, an altered MRI scan of a cancerous tumor might cause the AI model to incorrectly classify it as benign, resulting in a false negative and delaying critical treatment, thereby endangering the patient's life. Conversely, adversarial attacks may lead to false positives, erroneously identifying a healthy individual as having a serious illness, which could lead to unnecessary treatments, emotional distress, and increased healthcare costs.

Surveillance and security systems that utilize AI employ advanced DL models to analyze video feeds, recognize faces, detect anomalous behavior, and identify potential threats in real time. These models are trained on vast datasets containing images of people, objects, and actions, enabling them to differentiate between normal and suspicious activities. AI-powered security systems are deployed in high-stakes environments, including airports, banks, smart cities, and military bases, to enhance public safety by detecting threats more rapidly than human operators (Ardabili et al., 2023). However, evasion adversarial attacks can manipulate input images or video frames by subtly modifying pixel values, allowing the attacker to deceive the AI model without detection by human observers (Ahmed and Echi, 2021). For instance, an adversary might alter their facial features using adversarial perturbations, causing the system to mistakenly identify them as an authorized individual, thereby circumventing biometric authentication (dazeddigital, 2025). Similarly, tampering with an object, such as concealing a weapon, may result in the system failing to recognize the threat, leading to significant security breaches. In high-pressure environments, such as national security, banking, and critical infrastructure protection, these attacks could lead to unauthorized access, security threats, and potential harm to public safety.

6 DISCUSSION AND LIMITATIONS

Testing ML models with various architectural designs and features against adversarial samples generated by different techniques provides critical insights into the resilience and robustness of these models. In this study, five models were evaluated on image classification tasks using adversarial samples generated by four distinct techniques: the FGSM, PGD, DeepFool, and C&W. The findings revealed significant variation in the effectiveness of these techniques, which was influenced by the architectural differences of the models and the specific mechanisms employed by each attack. As a result, specific attack methods were more effective on some models than others, highlighting the nuanced relationship between attack characteristics and model architecture.

FGSM and PGD were more effective on simpler models, such as the FCNN and Simple CNN, causing a notable reduction in their performance. On the other hand, C&W, a more sophisticated attack, did not consistently produce the greatest performance degradation across all models. It appeared to be more effective against more complex models like MobileNetV2, which caused significant damage, whereas simpler architectures like FCNN were less affected. This discrepancy is likely because the C&W attack does not rely heavily on gradients, which are pivotal in the decision-making process of simpler models. The relative resilience of simpler models, like FCNN, under the C&W attack is particularly noteworthy, as these models exhibited pronounced vulnerability under the other attack techniques. The simplicity of their decision boundaries likely made them less susceptible to the highly optimized C&W attack. DeepFool, by contrast, demonstrated exceptional effectiveness across all tested models, successfully exploiting simple and complex architectures. However, the perturbations introduced by DeepFool were more visually perceptible to human observers, indicating a trade-off between the effectiveness of the attack and its subtlety.

The implications of these findings are significant for understanding the resilience and robustness of ML models. The varying effectiveness of different adversarial techniques underscores the importance

of selecting appropriate methods for evaluating and strengthening model robustness, particularly in applications where model reliability is critical. This study also emphasizes the need for tailored defense strategies against specific adversarial attacks. For instance, defenses against DeepFool may focus on enhancing boundary robustness, whereas defenses against C&W may prioritize detecting and correcting subtle noise in the input.

However, it is important to acknowledge the limitations of this study. First, the attack configurations were fixed, which may have constrained the full potential of some adversarial techniques. Parameters such as the step size for PGD or the optimization bounds for C&W were not fine-tuned for each model. Optimizing these parameters could enhance the effectiveness of the attacks, particularly on more advanced models such as MobileNetV2 and VGG11.

Furthermore, this study did not incorporate or evaluate any defense mechanisms commonly employed to mitigate and detect adversarial attacks Bountakas et al. (2023). On the one hand, regarding implementation of mitigation techniques such as: (i) *adversarial training* that improves the model's robustness by incorporating both benign and adversarial examples into the training dataset Goodfellow et al. (2025); Abou Khamis et al. (2020); (ii) *defensive distillation*, where a DNN learns using knowledge distilled from a more complex DNN, enabling the transfer of information to DL models with constrained computational resources Hinton et al. (2023); Papernot et al. (2016); (iii) *ensemble methods* where multiple ML algorithms are combined to solve a learning problem Apruzzese et al. (2020); Jiang et al. (2022), and (iv) *pre-processing* that encompasses existing works employing pre-processing techniques (e.g., feature reduction) to enhance model robustness against adversarial attacks Rosenberg et al. (2019); Xu et al. (2017). On the other hand, regarding detection techniques, the following techniques could be investigated: *supervised/semi-supervised learning* methods that detect adversarial ML, deploying an auxiliary ML/DL model that has been trained using supervised or semi-supervised learning Pawlicki et al. (2020); Taheri et al. (2020), and *distance-based* detection that is performed by measuring the distance between adversarial and original samples Meng and Chen (2017); Paudice et al. (2018). Integrating such defenses would provide a clearer understanding of the models' practical robustness in real-world applications where adversarial threats are prevalent.

7 RELATED WORK

This section provides an in-depth examination of prior works for the EvAIsion adversarial attacks and, offers a concise comparison of them against our work.

Villegas-Ch et al. (Villegas-Ch et al., 2024) investigated the effectiveness of three popular adversarial attack methods (i.e., FGSM, PGD, and C&W). Their experiments used image-classification models such as VGG16 and ResNet and reviewed the impact of adversarial perturbations on accuracy. Their study emphasized the model's vulnerability to adversarial inputs and proposed defensive strategies like image compression and Gaussian blurring to mitigate attacks. Furthermore, authors in (Ayub et al., 2020) performed a model evasion attack against the built MLP network for IDS using an adversarial ML technique known as JSMA method. Their experimental results show that the model evasion attack is capable of significantly reducing the accuracy of the IDS. Also, their findings supported that neural network-based IDS is susceptible to model evasion attacks, and attackers can essentially use this technique to evade IDS effectively.

Ahmed and Neeru (Ahmed and Neeru, 2024) analyzed gradient-based, decision-based, and optimization-based adversarial techniques. Its authors evaluated methods such as FGSM, DeepFool, C&W, and Zeroth Order Optimization (ZOO) across multiple models, including Logistic Regression, Support Vector Machines (SVM), and Gradient Boosting Classifiers. Their results include that Logistic Regression and SVM showed notable weaknesses in the face of all attack strategies, while Random Forest and Gradient Boosting models were more robust than simpler models.

Rahman et al. (Rahman et al., 2025) evaluated the robustness of several DNN models, such as ResNet152, VGG11, and DenseNet201 against four well-known adversarial attacks, FGSM, PGD, Basic Iterative Method (BIM), and DeepFool. The evaluation was performed on ImageNet, CIFAR-10, CIFAR-100 and SVHN data sets, while the authors proposed two ensemble adversarial attacks that combined three individual attacks. The ensemble methods were based on mean and weighted ensemble techniques and demonstrated a greater performance drop, namely 58%, compared to 39% of individual attacks.

Sen et al. (Sen and Dasgupta, 2023), employed the FGSM and adversarial patch attack on several CNN image classifiers, such as ResNet-34, GoogleNet, and DenseNet-161 to evaluate their robustness. To

Table 6. Comparison of EVAISION with existing works including the top performance reduction compared to baseline

Approach	Assessed Deep Learning Models	Adversarial Methods	Datasets	Reduction
Villegas-Ch et al. (2024)	VGG16, ResNet	FGSM, PGD, C&W	MNIST, CIFAR-10	35%
Ahmed and Neeru (2024)	Logistic Regression, SVM, Gradient Boosting	FGSM, DeepFool, C&W, ZOO	MNIST	98.47%
Rahman et al. (2025)	DNN (ResNet152, DenseNet201)	FGSM, PGD, DeepFool, BIM	ImageNet, CIFAR-10, CIFAR-100, SVHN	58%
Sen and Dasgupta (2023)	ResNet34, GoogleNet, DenseNet-161	FGSM, adversarial patch attack	ImageNet	97%
Hassan et al. (2022)	InceptionV3, AlexNet, ResNet18, VGG16	FGSM	ImageNet	99%
Sarkar et al. (2024)	ResNet50, DenseNet201, VGG19	FGSM, C&W	ImageNet	97%
EVAISION	FCNN, Simple CNN, LeNet, MobileNetV2, VGG111	FGSM, PGD, DeepFool, C&W	MNIST, Fashion-MNIST, CIFAR-10	96%

accomplish this, the ImageNet dataset was used and the results indicate that FGSM severely impacts the classification task, resulting in up to 97% classification error. Hassan et al. (Hassan et al., 2022) utilized FGSM attack on InceptionV3, AlexNet, ResNet18, and VGG16 CNN. The experiments performed on the ImageNet dataset and resulted in 99% misclassification accuracy.

In addition, in the postgraduation study that presented in (Sarkar et al., 2024), the authors explored the vulnerabilities of deep neural networks to adversarial attacks. Specifically, they evaluated the performance of three widely used CNN architectures, including ResNext50, DenseNet201, and VGG19, in adversarial examples generated by the FGSM and C&W. The authors also tested the defensive distillation method as a countermeasure to the FGSM and C&W attacks.

Table 6 compares EVAISION with existing works in terms of the assessed ML models, the adversarial methods implemented, the datasets deployed and the impacts of the adversarial methods in the deployed ML models. This current work goes beyond the works mentioned by conducting a comprehensive comparative analysis of four adversarial attacks, including FGSM, PGD, DeepFool, and C&W, on five machine learning models with diverse architectural designs (i.e., FCNN, Simple CNN, LeNet, MobileNetV2 and VGG11). In contrast to the aforementioned works, this work evaluates the impact of these attacks under uniform conditions using the MNIST, Fashion-MNIST, CIFAR-10 datasets. The evaluation methodology utilizes multiple performance metrics, including accuracy, precision, recall, F1 score, and mean confidence, to provide a more holistic perspective on model performance.

Apart from the aforementioned works, there is a notable contribution in literature focusing on Large Language Models (LLM) 's security that continues to evolve and be used in various topics, raising at the same time security threats. Aguilera-Martínez and Berzal (2025) presents a categorization of attacks targeting LLMs, distinguishing between those that occur during the training phase and those that target deployed models. The study also reviews corresponding defense strategies and evaluates the effectiveness of existing mechanisms against various security threats. Also, Ning et al. (2024) introduces the CheatAgent that leverages human-like capabilities of LLM to target LLM-powered RecSys. Specifically, it identifies the insertion position for maximum impact with minimal input modification then generates adversarial perturbations using an LLM-based agent. Moreover, authors in Xu et al. (2023) propose the PromptAttack to audit LLM's adversarial robustness, transforming textual adversarial attacks into crafted prompts, inducing the LLM to generate adversarial samples to fool itself. To ensure semantic consistency, a fidelity filter is used, while the attack strength is further enhanced by ensembling adversarial examples across varying levels of perturbation. Finally, Li et al. (2025) presents a taxonomy of attacks targeting agent-based systems, categorizing them by threat actors, objectives, entry points, attacker observability, attack strategies, and inherent vulnerabilities of agent pipelines. It is complemented by illustrative attacks on widely used open-source and commercial agents, highlighting the immediate and practical implications of these security weaknesses.

Furthermore, there is significant progress in game-theoretical approaches in adversarial ML. Particularly, Gao et al. (2023) provides an overview of existing game-theoretical approaches in adversarial ML for adaptively defending against adversarial attacks. To assess these methods, the authors proposed a set of evaluation metrics that highlight their respective strengths and limitations. Also, Dasgupta and Collins (2019) offers a comprehensive survey of techniques that enhance the robustness of ML algorithms against adversarial attacks through the application of game-theoretic frameworks. Finally, in Dritsoula et al. (2017), the authors model the interaction as a game between a defender who selects a classifier to distinguish between attacks and normal behavior based on a set of observed features.

8 CONCLUSION

This study presented a comparative analysis of the impact of four adversarial techniques (i.e., FGSM, PGD, DeepFool, and C&W) on five distinct models, each characterized by different architectural designs. Various performance metrics were employed to comprehensively assess how each model responded to adversarial samples. The effectiveness of the adversarial example generation techniques varied considerably depending on the model architecture. The FGSM attack notably compromised the performance of simpler models, such as FCNN and Simple CNN. Similarly, PGD primarily affected simpler architectures but exhibited a more pronounced impact on VGG11 when compared to FGSM. This suggests that the iterative nature of PGD allowed it to degrade the performance of even more complex models. On the other hand, the C&W technique, being a more advanced attack method, was expected to affect all models more severely. However, it caused a more substantial performance degradation in more complex models like MobileNetV2, while simpler models like FCNN experienced less impact than anticipated. In contrast, FGSM and PGD had a more significant effect on FCNN. DeepFool, however, demonstrated the most substantial effect across all tested models, with all performance metrics being notably degraded. Despite this, the adversarial samples generated by DeepFool were more easily distinguishable to the human eye, as the modifications it induced were more pronounced than those of other techniques. The fixed parameters in the adversarial attacks may have limited their full potential impact. Additionally, the study did not explore the effect of defensive strategies, such as adversarial training or input preprocessing. Future research incorporating these defense mechanisms could provide valuable insights into enhancing the resilience and robustness of models against adversarial attacks.

DECLARATIONS

Competing Interest Statement: The authors declare no conflict of interest.

Compliance with Ethical Standards: This article does not contain any studies with human participants or animals performed by any of the authors.

Generative AI and AI-Assisted Technologies in the Writing Process: During the preparation of this work, the authors used Grammarly to improve language and readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

Abou Khamis, R., Shafiq, M. O., and Matrawy, A. (2020). Investigating resistance of deep learning-based ids against adversaries using min-max optimization. In *ICC 2020-2020 IEEE international conference on communications (ICC)*, pages 1–7. IEEE.

Aguilera-Martínez, F. and Berzal, F. (2025). Llm security: Vulnerabilities, attacks, defenses, and countermeasures. *arXiv preprint arXiv:2505.01177*.

Ahmed, A. A. and Echi, M. (2021). Hawk-eye: An ai-powered threat detector for intelligent surveillance cameras. *Ieee Access*, 9:63283–63293.

Ahmed, A. A. and Neeru, N. (2024). A comparative analysis of adversarial attack methods on machine learning models. *International Journal of Scientific Research in Engineering and Management*, 8(008).

Algarni, A. and Thayananthan, V. (2025). Digital health: The cybersecurity for ai-based healthcare communication. *IEEE Access*.

Apruzzese, G., Andreolini, M., Marchetti, M., Colacino, V. G., and Russo, G. (2020). Appcon: Mitigating evasion attacks to ml cyber detectors. *Symmetry*, 12(4):653.

Ardabili, B. R., Pazho, A. D., Noghre, G. A., Neff, C., Bhaskararayuni, S. D., Ravindran, A., Reid, S., and Tabkhi, H. (2023). Understanding policy and technical aspects of ai-enabled smart video surveillance to address public safety. *Computational Urban Science*, 3(1):21.

Ayub, M. A., Johnson, W. A., Talbert, D. A., and Siraj, A. (2020). Model evasion attack on intrusion detection systems using adversarial machine learning. In *2020 54th annual conference on information sciences and systems (CISS)*, pages 1–6. IEEE.

Bhagoji, A. N., Cullina, D., Sitawarin, C., and Mittal, P. (2018). Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE.

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Bountakas, P., Zarras, A., Lekidis, A., and Xenakis, C. (2023). Defense strategies for adversarial machine learning: A survey. *Computer Science Review*, 49:100573.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee.

Charalambous, M., Farao, A., Kalantzantonakis, G., Kanakakis, P., Salamanos, N., Kotsifakos, E., and Froudakis, E. (2022). Analyzing coverages of cyber insurance policies using ontology. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–7.

Chowdhury, A., Karmakar, G., Kamruzzaman, J., Jolfaei, A., and Das, R. (2020). Attacks on self-driving cars and their countermeasures: A survey. *IEEE Access*, 8:207308–207342.

Chuah, J., Kruger, U., Wang, G., Yan, P., and Hahn, J. (2022). Framework for testing robustness of machine learning-based classifiers. *Journal of Personalized Medicine*, 12(8):1314.

csoonline (2025). Hacker allegedly puts massive OmniGPT breach data for sale on the dark web . `https://www.csoonline.com/article/3822911/hacker-allegedly-puts-massive-omnigpt-breach-data-for-sale-on-the-dark-web.html`.

Cui, J., Liew, L. S., Sabaliauskaite, G., and Zhou, F. (2019). A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks*, 90:101823.

Dasgupta, P. and Collins, J. (2019). A survey of game theoretic approaches for adversarial machine learning in cybersecurity tasks. *AI Magazine*, 40(2):31–43.

dazeddigital (2025). This ugly af t-shirt blocks facial recognition technology. `https://www.dazeddigital.com/science-tech/article/56745/1/could-sex-for-procreation-soon-be-obsolete`.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. (2018). Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193.

Dritsoula, L., Loiseau, P., and Musacchio, J. (2017). A game-theoretic analysis of adversarial classification. *IEEE Transactions on Information Forensics and Security*, 12(12):3094–3109.

Fang, C.-Y., Chen, S.-W., and Fuh, C.-S. (2003). Road-sign detection and tracking. *IEEE transactions on vehicular technology*, 52(5):1329–1341.

Farao, A., Ntantogian, C., Karagiannis, S., Magkos, E., Dritsa, A., and Xenakis, C. (2024). Nitro: an interconnected 5g-iot cyber range. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, pages 1–6.

Gao, L., Yan, Z., Liang, X., Xu, X., Wang, J., Ding, W., and Yang, L. T. (2023). Taxonomy and recent advance of game theoretical approaches in adversarial machine learning: A survey. *ACM Transactions on Sensor Networks*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). Deep learning cambridge. *MA: MIT Press http://www. deeplearningbook. org*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2025). explaining and harnessing adversarial examples. arxiv. org 2014.

Hassan, M., Younis, S., Rasheed, A., and Bilal, M. (2022). Integrating single-shot fast gradient sign method (fgsm) with classical image processing techniques for generating adversarial attacks on deep learning classifiers. In *Fourteenth International Conference on Machine Vision (ICMV 2021)*, volume 12084, pages 323–334. SPIE.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G., Vinyals, O., and Dean, J. (2023). Distilling the knowledge in a neural network." arxiv, mar. 09, 2015. *Accessed: Jun*, 11.

Hussain, M., Shang, Z., and Hong, J.-E. (2025). Adaptive precision layering for efficient adversarial training of deep learning models in intelligent vehicles. *Expert Systems with Applications*, page 126752.

IBM (2025). Adversarial Robustness Toolbox. `https://github.com/Trusted-AI/adversarial-robustness-toolbox`.

Jiang, H., Lin, J., and Kang, H. (2022). Fgmd: A robust detector against adversarial attacks in the iot

network. *Future Generation Computer Systems*, 132:194–210.

Joshi, S., Kataria, S., Shao, Y., Zelasko, P., Villalba, J., Khudanpur, S., and Dehak, N. (2022). Defense against adversarial attacks on hybrid speech recognition using joint adversarial fine-tuning with denoiser. *arXiv preprint arXiv:2204.03851*.

Kotyan, S. (2023). A reading survey on adversarial machine learning: Adversarial attacks and their understanding. *arXiv preprint arXiv:2308.03363*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. =https://api.semanticscholar.org/CorpusID:18268744.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, H., Bae, H., and Yoon, S. (2020). Gradient masking of label smoothing in adversarial robustness. *IEEE Access*, 9:6453–6464.

Li, A., Zhou, Y., Raghuram, V. C., Goldstein, T., and Goldblum, M. (2025). Commercial llm agents are already vulnerable to simple yet dangerous attacks. *arXiv preprint arXiv:2502.08586*.

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. (2017). Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*.

Linklaters (2024). The muah.ai data breach – Extortion threats and cyber vulnerabilities. `https://www.linklaters.com/en/insights/blogs/digilinks/2024/october/the-muah-ai-data-breach---extortion-threats-and-cyber-vulnerabilities`.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mahimai, L. D., Gunasekaran, M., Kim, J., and Kadry, S. (2025). Adversarial robustness enhancement in deep learning-based breast cancer classification: A multi-faceted approach to poisoning and evasion attack mitigation. *Alexandria Engineering Journal*, 115:65–82.

Meng, D. and Chen, H. (2017). Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 135–147.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ning, L.-b., Wang, S., Fan, W., Li, Q., Xu, X., Chen, H., and Huang, F. (2024). Cheatagent: Attacking llm-empowered recommender systems via llm agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2284–2295.

Pantelakis, V., Bountakas, P., Farao, A., and Xenakis, C. (2023). Adversarial machine learning attacks on multiclass classification of iot network traffic. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, pages 1–8.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE.

Paudice, A., Muñoz-González, L., Gyorgy, A., and Lupu, E. C. (2018). Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv preprint arXiv:1802.03041*.

Pawlicki, M., Choraś, M., and Kozik, R. (2020). Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, 110:148–154.

Petihakis, G., Farao, A., Bountakas, P., Sabazioti, A., Polley, J., and Xenakis, C. (2024). Aias: Ai-assisted cybersecurity platform to defend against adversarial ai attacks. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, pages 1–7.

PyTorch Foundation (2025). torchvision. `https://pytorch.org/vision/stable/index.htmll`.

Rahman, M., Roy, P., Frizell, S., and Qian, L. (2025). Evaluating pretrained deep learning models for image classification against individual and ensemble adversarial attacks. *IEEE Access*.

Rosenberg, I., Shabtai, A., Elovici, Y., and Rokach, L. (2019). Defense methods against adversarial examples for recurrent neural networks. *arXiv preprint arXiv:1901.09963*.

Sarkar, T. R., Das, N., Maitra, P. S., Some, B., Saha, R., Adhikary, O., Bose, B., and Sen, J. (2024). Evaluating adversarial robustness: A comparison of fgsm, carlini-wagner attacks, and the role of distillation as defense mechanism. *arXiv preprint arXiv:2404.04245*.

Sen, J. and Dasgupta, S. (2023). Adversarial attacks on image classification models: Fgsm and patch attacks and their impact. *arXiv preprint arXiv:2307.02055*.

Shaheen, M. Y. (2021). Applications of artificial intelligence (ai) in healthcare: A review. *ScienceOpen Preprints*.

Sharma, N. and Kaushik, P. (2025). Integration of ai in healthcare systems—a discussion of the challenges and opportunities of integrating ai in healthcare systems for disease detection and diagnosis. *AI in Disease Detection: Advancements and Applications*, pages 239–263.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Suciu, G., Farao, A., Bernardinetti, G., Palamà, I., Sachian, M.-A., Vulpe, A., Vochin, M.-C., Muresan, P., Bampatsikos, M., Muñoz, A., and Xenakis, C. (2022). Samgrid: security authorization and monitoring module based on sealedgrid platform. *Sensors*, 22(17):6527.

Szegedy, C. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Taheri, R., Javidan, R., Shojafar, M., Pooranian, Z., Miri, A., and Conti, M. (2020). On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications*, 32(18):14781–14800.

The Linux Foundation (2025). PyTorch. `https://pytorch.org/`.

The Register (2025). Cheap 'n' simple sign trickery will bamboozle self-driving cars, fresh research claims. `https://www.theregister.com/2025/03/07/lowcost_malicious_attacks_on_selfdriving/`.

TIME (2025). When AI Thinks It Will Lose, It Sometimes Cheats, Study Finds. `https://time.com/7259395/ai-chess-cheating-palisade-research/`.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. (2016). Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618.

Villegas-Ch, W., Jaramillo-Alcázar, A., and Luján-Mora, S. (2024). Evaluating the robustness of deep learning models against adversarial attacks: An analysis with fgsm, pgd and cw. *Big Data and Cognitive Computing*, 8(1):8.

Wang, Y., Sun, T., Li, S., Yuan, X., Ni, W., Hossain, E., and Poor, H. V. (2023). Adversarial attacks and defenses in machine learning-empowered communication systems and networks: A contemporary survey. *IEEE Communications Surveys & Tutorials*.

Wu, B., Wei, S., Zhu, M., Zheng, M., Zhu, Z., Zhang, M., Chen, H., Yuan, D., Liu, L., and Liu, Q. (2023). Defenses in adversarial machine learning: A survey. *arXiv preprint arXiv:2312.08890*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, W., Evans, D., and Qi, Y. (2017). Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*.

Xu, X., Kong, K., Liu, N., Cui, L., Wang, D., Zhang, J., and Kankanhalli, M. (2023). An llm can fool itself: A prompt-based adversarial attack. *arXiv preprint arXiv:2310.13345*.

Zhang, K., Cheng, S., Shen, G., Ribeiro, B., An, S., Chen, P.-Y., Zhang, X., and Li, N. (2025). Censor: Defense against gradient inversion via orthogonal subspace bayesian sampling. *arXiv preprint arXiv:2501.15718*.