ARAIC

2022 Applied Research in Artificial Intelligence and Cloud Computing

A COMPARATIVE ANALYSIS OF ADVERSARIAL CAPABILITIES, ATTACKS, AND DEFENSES ACROSS THE MACHINE LEARNING PIPELINE IN WHITE-BOX AND BLACK-BOX SETTINGS

Tasneem Hossain

Abstract

The increasing adoption of machine learning models across various domains has brought to light the critical issue of their vulnerability to adversarial attacks, raising concerns about their security and reliability. This research discusses the adversarial capabilities that can be exploited at different stages of the machine learning pipeline: *training, testing, and deployment*. We investigate the distinct challenges and opportunities adversaries face in both transparent (white-box) and opaque (black-box) settings. Adversaries can tamper with the training data during the initial phase of the machine learning pipeline, compromising the model's learning process. In a transparent setup, adversaries possess the ability to directly alter, introduce, or eliminate samples, injecting malicious patterns. Conversely, in an opaque setup, adversaries can indirectly sway the training process by manipulating data collection or preprocessing stages. Safeguarding against training-stage attacks necessitates data cleansing, anomaly identification, and resilient training techniques like adversarial training. Moving on to the testing phase, adversaries concentrate on designing deceptive examples that mislead the trained model. Adversaries with comprehensive knowledge of the model, operating in a white-box scenario, can meticulously craft highly targeted adversarial instances. On the other hand, black-box adversaries, lacking direct access to the model, employ techniques such as transferability to generate adversarial examples. Effective countermeasures against testing-stage attacks encompass adversarial training, ensemble approaches, and randomized defense mechanisms. As the model is deployed and used in real-world contexts, adversaries exploit vulnerabilities to undermine its performance. In a white-box setting, adversaries can meticulously examine the model's behavior and engineer targeted attacks. Conversely, black-box adversaries probe the model and exploit weaknesses by carefully constructing malicious inputs. To protect against deployment-stage threats, defenses such as real-time monitoring, anomaly detection, secure deployment practices, and regular security assessments are also discussed.

Applied Research in Artificial Intelligence and Cloud Computing

1. Introduction

Machine learning has bacame a transformative technology in the field of artificial intelligence, enabling computers to achieve remarkable performance across various tasks. From image classification and speech recognition to machine translation and game-playing, machine learning systems have demonstrated an extraordinary ability to learn from data and make intelligent decisions. These advancements have opened up new possibilities in numerous domains, impacting industries such as healthcare, finance, transportation, and entertainment.

However, despite the impressive successes

of machine learning models, a significant vulnerability has been discovered: adversarial examples Dong et al., 2018. Adversarial examples are carefully crafted inputs that are specifically designed to deceive machine learning classifiers Lin et al., 2017. An attacker can create adversarial examples that are misclassified by the learned model, leading to incorrect predictions or decisions by introducing imperceptible perturbations to the original inputs,. Szegedy et al., 2013 introduced the idea of adversarial examples, which revealed a significant vulnerability in neural networks. Their research has generated considerable interest among scientists studying adversarial attacks. It is expected that the number and types of adversarial attacks will keep growing in the future, driven by the increasing potential for economic benefits. This trend presents both challenges and opportunities for creating AI systems that are robust and resilient against such attacks Madry et al., 2017.

What makes adversarial examples alarming is that they are often indistinguishable from the original inputs to the human eye. The perturbations added to the input data are so subtle that they go unnoticed by humans, yet they are sufficient to fool the machine learning model. This highlights a fundamental difference between how humans and machines process and interpret information, raising questions about the robustness and reliability of machine learning systems in real-world scenarios Bradshaw et al., 2017.

The existence of adversarial examples poses significant challenges and risks in various applications of machine learning. In safety-critical domains such as autonomous vehicles or medical diagnosis, the consequences of a machine learning model being deceived by an adversarial example can be severe. For instance, an adversarial perturbation added to a stop sign could cause an autonomous vehicle to misinterpret it as a speed limit sign, leading to dangerous behavior on the road Cai et al., 2018. Similarly, in medical imaging, an adversarial example could trick a diagnostic model into misclassifying a malignant tumor as benign, potentially delaying necessary treatment.

Adversarial examples can be exploited as a means of malicious attack against machine learning systems. Attackers can take advantage of this vulnerability to manipulate the behavior of machine learning models for their own benefit or to cause harm. For example, an attacker could generate adversarial examples to bypass a facial recognition system, gaining unauthorized access to restricted areas or resources. In the tasks of spam filters or malware detection, adversarial examples could be crafted to evade detection, allowing malicious content to infiltrate systems undetected Dong et al., 2018.

The phenomenon of adversarial examples also highlights the need for a deeper understanding of how machine learning models make decisions and what they actually learn from the training data. It raises questions about the interpretability and explainability of these models, as well as their ability to generalize to new and unseen examples. The lack of transparency in the decision-making process of machine learning models can make it difficult to identify and address potential weaknesses or biases.

The existence of adversarial examples shows the importance of robustness and security in machine learning systems. As these systems become more and more integrated into critical applications and decision-making processes, ensuring their resilience against adversarial attacks is necessary. The vulnerability exposed by adversarial examples emphasizes the need for rigorous testing, validation, and monitoring of machine learning models to identify and mitigate potential risks Huang et al., 2017.

2. Adversarial capabilities, attacks, and defenses during training stage

In the training stage, adversaries employ various techniques to exploit vulnerabilities and undermine the model's learning process.

In a white-box setting, the adversary has full

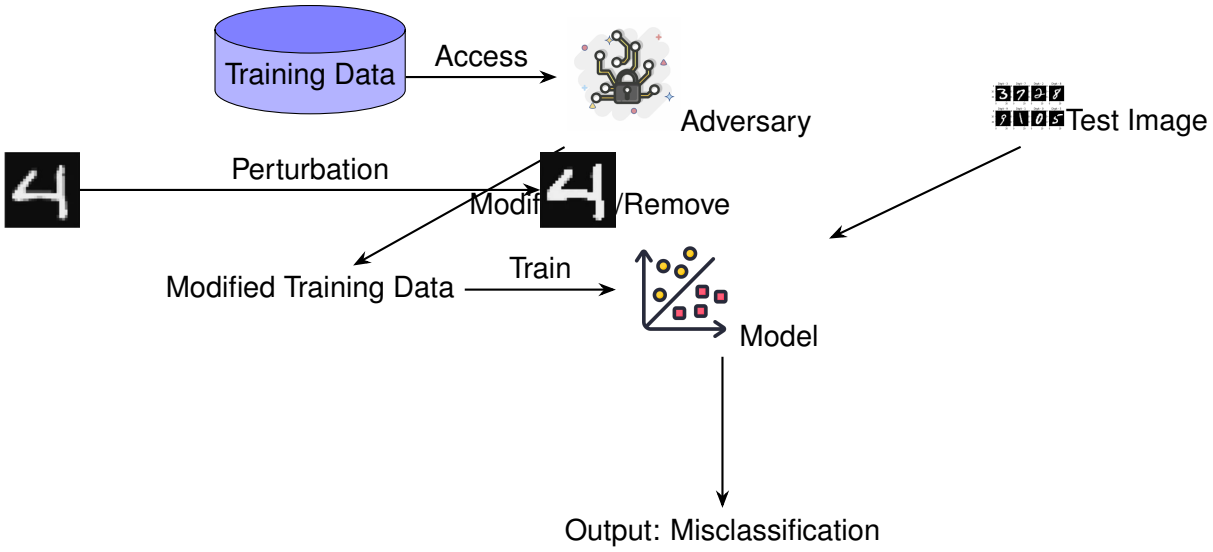

Figure 1: White-box adversarial attack on a machine learning model.

access to the training data and can directly modify, add, or remove examples from the dataset. This level of access empowers the adversary to craft malicious examples specifically designed to mislead the model during training. The adversary can manipulate the model by carefully perturbing existing examples or injecting new ones to learn incorrect patterns, biases, or behaviors that align with their intent Ilyas et al., 2018 Dai et al., 2018.

One common technique in white-box attacks is *data poisoning*. The adversary strategically modifies a subset of the training examples to introduce subtle perturbations or mislabeled instances. These manipulated examples are carefully crafted to be visually similar to benign examples but are assigned incorrect labels or contain imperceptible perturbations. When the model is trained on this poisoned dataset, it learns to associate the malicious patterns with the wrong labels, leading to misclassification or unintended behavior Lin et al., 2017 Liu et al., 2018.

For instance, in an image classification task where the model is trained to distinguish between different objects. An adversary could select a subset of images from the training set and add imperceptible noise or modify a few pixels in each image. These perturbed images are then mislabeled as a different object class. During training, the model is repeatedly exposed to these manipulated examples, causing it to learn the adversary's desired misclassification. As a result, when the trained model is presented with similar perturbed images during inference, it is likely to misclassify them, falling victim to the adversary's attack.

The effectiveness of white-box attacks lies in the adversary's ability to carefully craft the malicious examples based on their knowledge of the model's architecture, training process, and the specific dataset. Using this information, the adversary can optimize the perturbations to maximize the impact on the model's learning while minimizing the chances of detection.

Another technique used in white-box attacks is *gradient-based manipulation*. In this approach, the adversary exploits the model's gradient information to craft adversarial examples. The adversary can determine the direction and magnitude of perturbations that would most significantly impact the model's predictions by calculating the gradients of the model's loss function with respect to the input features. The adversary can generate adversarial examples that are specifically tailored to deceive the model by

iteratively adjusting the input features based on the gradients.

Gradient-based attacks can be effective when the adversary has access to the model's architecture and can compute the gradients during the training process. The adversary can guide the model's learning towards malicious patterns or behaviors by carefully manipulating the gradients. This can lead to the model learning incorrect decision boundaries or becoming overly sensitive to specific features that the adversary has designed Liu et al., 2018 Bradshaw et al., 2017. It is to note that white-box attacks require a high level of access and knowledge, which may not always be feasible in real-world scenarios. However, the potential consequences of such attacks cannot be overlooked in critical applications where the integrity and reliability of the model's predictions are of utmost importance.

In contrast to white-box attacks, black-box attacks operate under more restricted conditions. In a black-box setting, the adversary does not have direct access to the training data or the model's internal workings. However, they can still attempt to influence the training process indirectly by manipulating the data that is fed into the model.

One approach in black-box attacks is to *exploit weaknesses* in the data collection or *preprocessing pipeline*. Adversaries can try to inject malicious data points into the system, aiming to pollute the training dataset. This can be achieved by various means, such as submitting manipulated user-generated content, exploiting vulnerabilities in data scraping techniques, or compromising data sources.

For example, consider a scenario where an adversary targets a sentiment analysis model trained on user reviews. The adversary could generate a large number of fake reviews containing misleading or biased content and submit them to the data collection system. If the data preprocessing steps fail to filter out these malicious reviews effectively, they can end up in the training dataset, influencing the model's learning process. As a result, the trained model may exhibit biased or incorrect predictions when applied to real-world data Madry et al., 2017. Black-box attacks often rely on the adversary's ability to generate adversarial examples that can deceive the model. Even without access to the training data, adversaries can craft perturbations or manipulate inputs in a way that exploits the model's vulnerabilities. The adversary aims to mislead the model and cause it to make incorrect predictions by carefully designing these adversarial examples and injecting them into the data stream.

One technique used in black-box attacks is *transferability-based attacks*. In this approach, the adversary trains a substitute model using a dataset similar to the target model's training data. The adversary can generate adversarial examples that are likely to transfer to the target model by querying the target model with carefully crafted inputs and observing its outputs. These transferable adversarial examples can then be used to attack the target model, even without direct access to its training data or architecture.

The success of black-box attacks depends on various factors, including the robustness of the data collection and preprocessing pipeline, the model's architecture and training process, and the adversary's ability to generate effective adversarial examples. While black-box attacks may not have the same level of precision and control as white-box attacks, they still pose a significant threat, especially in scenarios where the model relies on external data sources or user-generated content.

Another aspect of adversarial attacks during training is the concept of backdoor attacks. In a backdoor attack, the adversary manipulates the training data by injecting a specific trigger or pattern that is associated with a desired malicious behavior. The trigger can be a particular pixel pattern, a specific word or phrase, or any other

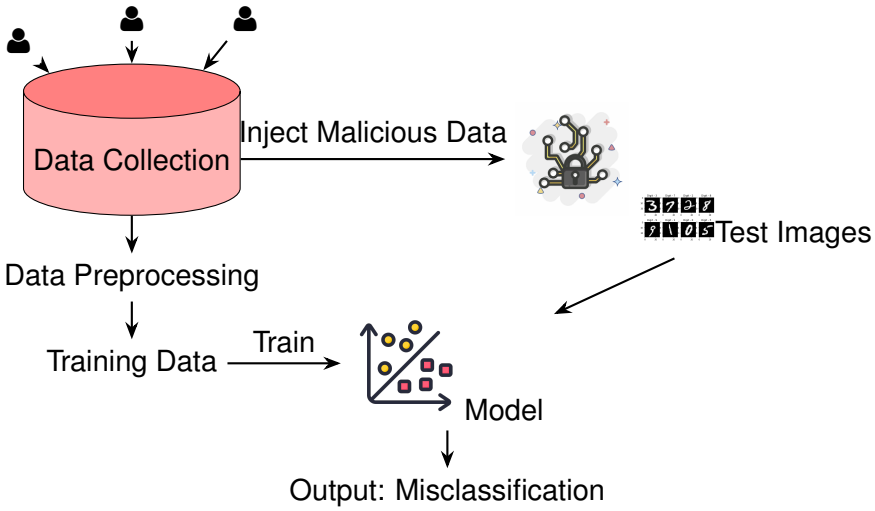

Figure 2: Black-box adversarial attack on a machine learning model.

carefully designed feature. During training, the model learns to associate the trigger with the malicious behavior, effectively creating a backdoor that the adversary can exploit later.

For instance, an adversary could manipulate a subset of the training images by adding a specific pixel pattern or a small object, such as a sticker, to the faces. These manipulated images are then labeled as belonging to a particular authorized individual. During training, the model learns to associate the presence of the trigger with the authorized label. As a result, when the trained model is deployed, the adversary can gain unauthorized access by presenting an image with the trigger, bypassing the system's security measures.

Backdoor attacks can be challenging to detect and mitigate because the trigger is typically designed to be inconspicuous and not easily noticeable to human observers. The model's behavior appears normal for benign inputs, but the presence of the trigger activates the malicious behavior. This makes backdoor attacks a serious concern, especially in sensitive applications where the model's predictions have significant consequences Narodytska and Kasiviswanathan, 2017.

It is to recognize that adversarial attacks during training can have far-reaching implications beyond the immediate impact on the model's performance. Adversarially trained models can perpetuate biases, make unfair decisions, or be exploited for malicious purposes. In domains such as healthcare, finance, or criminal justice, where machine learning models are increasingly being used to make critical decisions, the consequences of adversarial attacks in this stage can be severe and have real-world impacts on individuals and society as a whole.

Moreover, adversarial attacks during training raise concerns about the trustworthiness and reliability of machine learning models. If models can be easily manipulated or deceived by adversaries, it undermines their credibility and poses risks to the users who rely on their predictions. This highlights the need for robust measures to detect and defend against adversarial attacks, as well as for increased transparency and accountability in the development and deployment of machine learning systems.

Adversarial attacks during the training stage pose a significant threat to the integrity and performance of machine learning models. Whether it is through white-box attacks, where the adversary has full access to the training data, or black-box attacks, where the adversary manipulates the input data stream, these attacks can

Technique	Dfenitions
Data Sanitization	$D' = \{x_i \in D \| x_i$ is not suspicious or malicious$\}$
Anomaly Detection	$A(x_i) = d(\phi(x_i), \phi(D'))$, where d is a distance metric and $\phi(D')$ represents the normal patterns in the sanitized dataset. Flag x_i as an anomaly if $A(x_i) > \tau$, where τ is a predefined threshold.
Adversarial Training	Generate adversarial examples x_i^{adv} for each $x_i \in D'$ using an attack method. Train the model f_θ on the augmented dataset $D^{adv} = \{(x_i, y_i), (x_i^{adv}, y_i) \| x_i \in D', y_i \in Y\}$. Update model parameters θ to minimize the loss function $\mathcal{L}(f_\theta(x_i), y_i) + \mathcal{L}(f_\theta(x_i^{adv}), y_i)$.
Training with Noisy Examples	Generate noisy examples x_i^{noisy} for each $x_i \in D'$ by adding random noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Train the model f_θ on the augmented dataset $D^{noisy} = \{(x_i, y_i), (x_i^{noisy}, y_i) \| x_i \in D', y_i \in Y\}$. Update model parameters θ to minimize the loss function $\mathcal{L}(f_\theta(x_i), y_i) + \mathcal{L}(f_\theta(x_i^{noisy}), y_i)$.

Table 1: Techniques for Defending Against Adversarial Attacks During Training

lead to models learning incorrect patterns, exhibiting biased behavior, or making erroneous predictions. The consequences of adversarial attacks extend beyond the immediate impact on the model's performance, as they can perpetuate biases, make unfair decisions, and undermine the trustworthiness of machine learning systems.

Defending against adversarial attacks during the training stage involves several techniques. Data sanitization is one such step, where the training data is carefully examined and filtered to remove any suspicious or potentially malicious examples. Let D be the training dataset and apply a function $f(D) \rightarrow D'$ to filter out suspicious or malicious examples, resulting in a sanitized dataset $D' = \{x_i \in D \mid x_i$ is not suspicious or malicious$\}$.

Anomaly detection methods can be employed to identify and flag unusual patterns or outliers in the training data. Let x_i be a training example and $\phi(x_i)$ be a function that extracts relevant features. Define an anomaly score $A(x_i) = d(\phi(x_i), \phi(D'))$, where d is a distance metric and $\phi(D')$ represents the normal patterns in the sanitized dataset. Flag x_i as an anomaly if $A(x_i) > \tau$, where τ is a predefined threshold.

Additionally, robust training methods, such as adversarial training or training with noisy examples, can help the model become more resilient to adversarial perturbations. For adversarial training, generate adversarial examples x_i^{adv} for each $x_i \in D'$ using an attack method and train the model f_θ on the augmented dataset $D^{adv} = \{(x_i, y_i), (x_i^{adv}, y_i) \mid x_i \in D', y_i \in Y\}$. Update model parameters θ to minimize the loss function $\mathcal{L}(f_\theta(x_i), y_i) + \mathcal{L}(f_\theta(x_i^{adv}), y_i)$. For training with noisy examples, generate noisy examples x_i^{noisy} for each $x_i \in D'$ by adding random noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and train the model f_θ on the augmented dataset $D^{noisy} = \{(x_i, y_i), (x_i^{noisy}, y_i) \mid x_i \in D', y_i \in Y\}$. Update model parameters θ to minimize the loss function $\mathcal{L}(f_\theta(x_i), y_i) + \mathcal{L}(f_\theta(x_i^{noisy}), y_i)$.

When incorporating these techniques, the model learns to correctly classify adversarial examples, reducing its vulnerability to such attacks. The training process becomes $\theta^* = \arg\min_\theta \mathbb{E}_{(x_i, y_i) \in D^{adv} \cup D^{noisy}}[\mathcal{L}(f_\theta(x_i), y_i)]$, where θ^* represents the optimal model parameters that minimize the expected loss over the augmented training dataset. Explicitly incorporating adversarial examples into the training process helps the model learn to correctly classify them, reducing its vulnerability to such attacks.

3. Adversarial capabilities, attacks, and defenses in testing stage

Adversarial attacks on machine learning models have also became a significant concern during the testing stage when the model is exposed to real-world data. These attacks aim to deceive the trained model by crafting carefully designed inputs known as *adversarial examples*, which can cause the model to make incorrect predictions or classifications.

At the core of adversarial attacks during the testing stage lies the concept of adversarial examples. These are strategically manipulated inputs that are specifically crafted to fool the model while appearing normal or innocuous to human observers. Adversaries can exploit the model's vulnerabilities and trigger misclassifications or unintended behaviors by introducing subtle perturbations or modifications to the input data. The severity of adversarial attacks is amplified by the fact that they can be difficult to detect and defend against. Adversarial examples often resemble legitimate inputs and can be imperceptible to the human eye. This makes it challenging to distinguish between benign and malicious instances, as the perturbations are carefully designed to be minimal yet effective in deceiving the model Narodytska and Kasiviswanathan, 2017 Pang et al., 2018. Adversarial attacks during the testing stage can be broadly categorized into two settings: white-box attacks and black-box attacks. The distinction lies in the level of access and knowledge the adversary possesses about the target model.

In a white-box setting, the adversary has complete access to the model's architecture, parameters, and even the test data. This knowledge empowers the adversary to perform a detailed analysis of the model's internals, uncovering specific vulnerabilities and weaknesses that can be exploited. The adversary can craft highly targeted adversarial examples tailored to the specific model by leveraging this information. White-box attacks often involve a process of studying the model's decision boundaries, activation patterns, and feature representations. The adversary can iteratively modify input features, adding imperceptible noise or applying carefully calculated perturbations to manipulate the model's output. These perturbations are designed to push the input across the decision boundary, causing the model to misclassify the example. One of the most prominent techniques used in white-box attacks is *gradient-based optimization*. Accessing the model's gradients allows the adversary to determine the direction and magnitude of perturbations that would have the most significant impact on the model's predictions. Through iterative optimization, the adversary can refine the adversarial examples to maximize their effectiveness while minimizing the perceptibility of the modifications Bradshaw et al., 2017 Pang et al., 2018. White-box attacks pose a significant threat because they allow adversaries to create highly customized and potent adversarial examples. These examples are crafted with a deep understanding of the model's weaknesses and can be extremely difficult to detect or defend against. In domains like autonomous vehicles, medical diagnosis, or financial systems, the consequences of white-box attacks are severe, where the model's predictions have critical implications.

In contrast to white-box attacks, black-box attacks operate under more limited access to the target model. In a black-box setting, the adversary does not have direct access to the model's internal details but can still interact with it by querying the model with inputs and observing the corresponding outputs. Despite the restricted access, black-box attacks can still pose significant risks to the model's integrity. One of the key techniques used in black-box attacks is *transferability*. Transferability refers to the property where adversarial examples generated for one model can also fool other models with similar architectures or trained on similar data. By exploiting this property, adversaries can create adversarial examples without direct access to the target model. The process of launching a black-box attack often involves training a surrogate model that mimics the behavior of the tar-

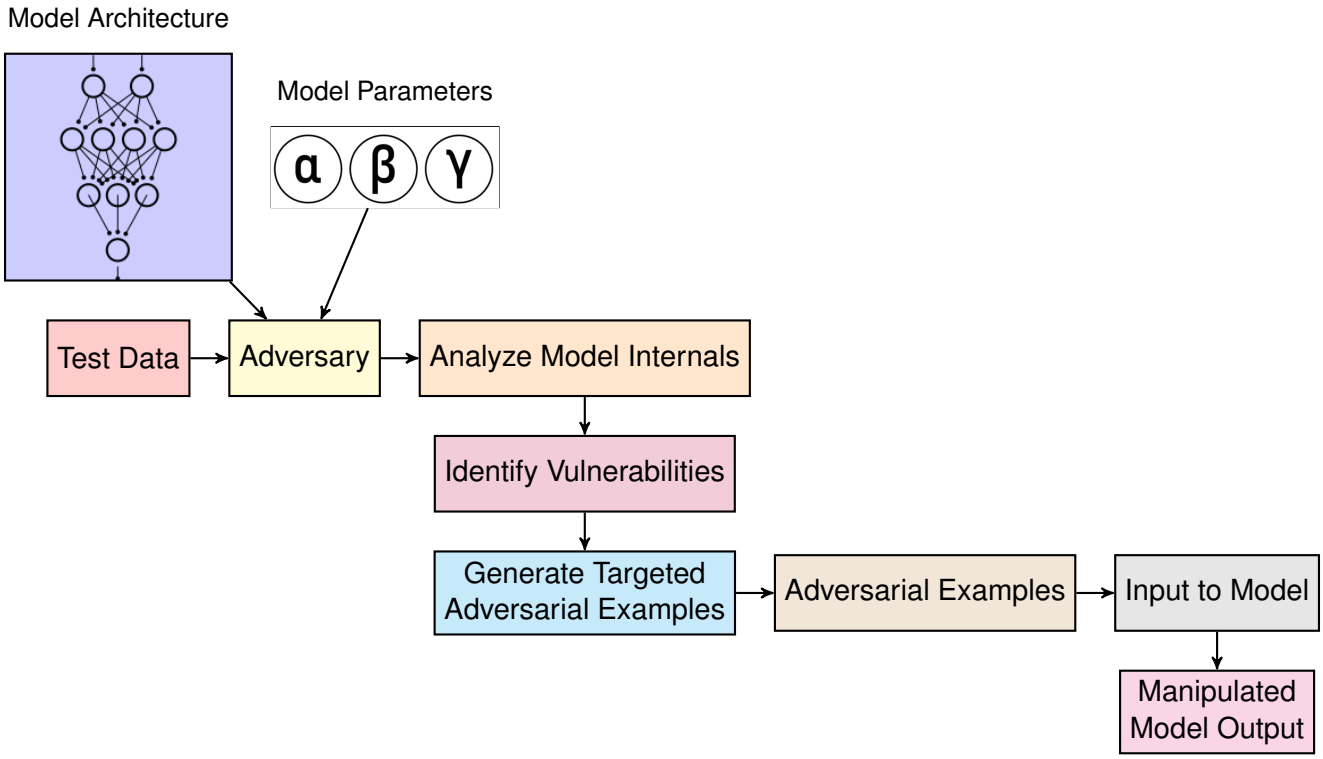

Figure 3: White-box adversarial attack process

get model. The adversary collects a dataset of inputs and their corresponding outputs by querying the target model. This dataset is then used to train the surrogate model, which serves as an approximation of the target model's decision boundaries. Once the surrogate model is trained, the adversary can generate adversarial examples using various techniques, such as gradient-based optimization or evolutionary algorithms. These adversarial examples are crafted to fool the surrogate model and, due to transferability, are likely to also deceive the target model. Black-box attacks highlight the importance of model robustness and the need for defenses that can withstand adversarial examples generated from different models or architectures. The transferability property shows the possibility for adversarial examples to have a broader impact, affecting not only the specific model they were crafted for but also other models in the same domain. The consequences of adversarial attacks during the testing stage can be significant and far-reaching. In safety-critical applications, such as autonomous vehicles or medical diagnosis, a single misclassification induced by an adversarial example can lead to disastrous outcomes. False positives or false negatives caused by adversarial attacks can undermine the reliability and trustworthiness of these systems, eroding public confidence in their deployment Huang et al., 2017 Pang et al., 2018.

Defending against adversarial attacks during the testing stage involves various techniques. One effective approach is adversarial training, where the model is explicitly trained on adversarial examples alongside the original clean examples. Let $D = \{(x_i, y_i)\}_{i=1}^{N}$ be the training dataset, where x_i is an input example and y_i is its corresponding label. Generate adversarial examples x_i^{adv} for each $x_i \in D$ using an attack method and train the model f_θ on the augmented dataset $D^{adv} = \{(x_i, y_i), (x_i^{adv}, y_i)\}_{i=1}^{N}$. Update model parameters θ to minimize the loss function: $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}[\ell(f_\theta(x_i), y_i) + \ell(f_\theta(x_i^{adv}), y_i)]$ where ℓ is a suitable loss function (e.g., cross-entropy loss).

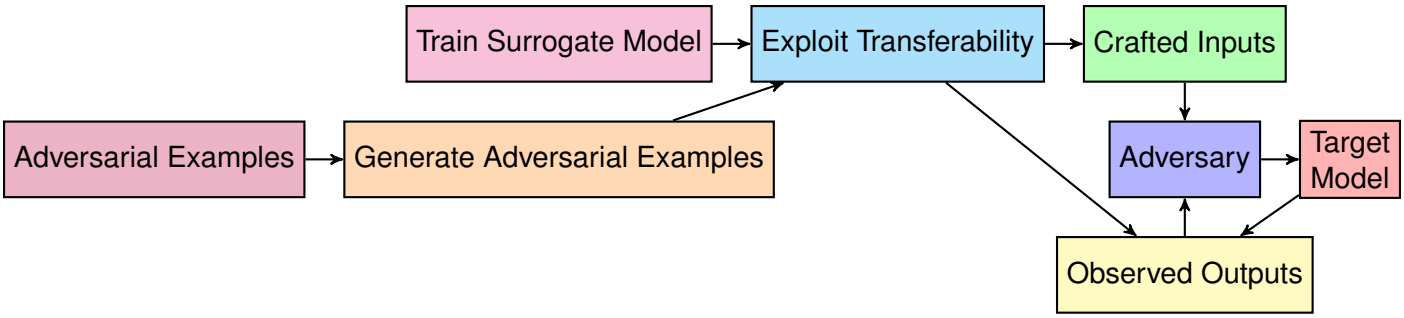

Figure 4: Black-box adversarial attack using transferability

Ensemble methods, where multiple models are combined to make predictions, can also improve robustness by leveraging the collective decision-making of different models. Let $\{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_M}\}$ be an ensemble of M trained models. For a test input x, obtain the predictions from each model: $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M\}$ and combine the predictions using a suitable aggregation function g: $\hat{y} = g(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M)$ where g can be majority voting, averaging, or weighted averaging based on model confidences Song et al., 2018.

Randomized defenses, such as adding random noise or applying random transformations to the input data, can make it harder for adversaries to craft effective adversarial examples. For random noise addition, generate a noisy version $x^{noisy} = x + \epsilon$ for a test input x, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is random Gaussian noise, and predict the label for x^{noisy} using the trained model: $\hat{y} = f_\theta(x^{noisy})$. For random transformations, define a set of random transformations $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ (e.g., rotation, translation, scaling), randomly select a transformation $T \in \mathcal{T}$ for a test input x, apply it to obtain $x^{transformed} = T(x)$, and predict the label for $x^{transformed}$ using the trained model: $\hat{y} = f_\theta(x^{transformed})$.

Incorporating these techniques during testing makes the model more robust to adversarial attacks. The prediction process can be formalized as: $\hat{y} = f_\theta(x^{adv})$ or $\hat{y} = g(f_{\theta_1}(x), f_{\theta_2}(x), \ldots, f_{\theta_M}(x))$ or $\hat{y} = f_\theta(x^{noisy})$ or $\hat{y} = f_\theta(x^{transformed})$, where $\hat{y}$ represents the predicted label for the test input x, obtained using one of the defensive techniques mentioned above. The model's robustness to adversarial attacks can be significantly improved by exposing it to adversarial examples during training and employing ensemble methods and randomized defenses during testing.

4. Adversarial capabilities, attacks, and defenses deployment stage

Adversarial attacks on machine learning models have emerged as a significant concern, particularly during the deployment stage when models are actively used to make decisions and predictions in real-world scenarios. These attacks exploit vulnerabilities in the deployed models, allowing adversaries to manipulate the model's behavior and cause unintended consequences Tramèr et al., 2017. The deployment stage presents unique challenges and risks, as the model is exposed to a wide range of inputs and interactions, making it susceptible to various types of adversarial attacks.

White-Box Attacks in Deployment: In a white-box setting, the adversary has extensive knowledge and access to the deployed model, including its architecture, parameters, and even the input data being fed into the model. This level of access enables the adversary to perform a comprehensive analysis of the model's behavior and identify potential vulnerabilities that can be exploited.

One of the key advantages of a white-box

Technique	Mathematical Expression
Adversarial Training	Generate adversarial examples x_i^{adv} for each $x_i \in D$ using an attack method. Train the model f_θ on the augmented dataset $D^{adv} = \{(x_i, y_i), (x_i^{adv}, y_i)\}_{i=1}^N$. Update model parameters θ to minimize the loss function: $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^N [\ell(f_\theta(x_i), y_i) + \ell(f_\theta(x_i^{adv}), y_i)]$ where ℓ is a suitable loss function (e.g., cross-entropy loss).
Ensemble Methods	Let $\{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_M}\}$ be an ensemble of M trained models. For a test input x, obtain the predictions from each model: $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M\}$. Combine the predictions using a suitable aggregation function g: $\hat{y} = g(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M)$ where g can be majority voting, averaging, or weighted averaging based on model confidences.
Random Noise Addition	For a test input x, generate a noisy version $x^{noisy} = x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is random Gaussian noise. Predict the label for x^{noisy} using the trained model: $\hat{y} = f_\theta(x^{noisy})$.
Random Transformations	Define a set of random transformations $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ (e.g., rotation, translation, scaling). For a test input x, randomly select a transformation $T \in \mathcal{T}$ and apply it to obtain $x^{transformed} = T(x)$. Predict the label for $x^{transformed}$ using the trained model: $\hat{y} = f_\theta(x^{transformed})$.

Table 2: Techniques for Defending Against Adversarial Attacks During Testing

attack during deployment is the ability to craft highly targeted and effective adversarial examples. By leveraging their knowledge of the model's internals, adversaries can design inputs that are specifically tailored to deceive the model and cause it to make incorrect predictions or decisions. These adversarial examples can be carefully constructed to exploit the model's weaknesses, such as its sensitivity to certain features or its reliance on specific patterns in the input data.

Figure 5 illustrates a white-box adversarial setting in the context of a self-driving car scenario. In this example, the adversary has access to the model architecture, parameters, and input data being used by the self-driving car's decision-making system. With this level of access, the adversary can analyze the model's behavior and identify vulnerabilities that can be exploited to manipulate the car's decisions in real-time.

For instance, the adversary could manipulate the input data, such as the images captured by the car's cameras or the sensor readings, to mislead the model and cause the car to make dangerous or unintended maneuvers. By carefully crafting perturbations or adding imperceptible noise to the input data, the adversary can exploit the model's sensitivity to specific features and trigger misclassifications or erroneous predictions.

The consequences of white-box attacks during deployment can also be severe in safety-critical applications like self-driving cars. A successful attack could compromise the system's integrity, leading to accidents, collisions, or other harmful outcomes. The adversary's ability to manipulate the model's behavior in real-time poses significant risks to the safety and reliability of the deployed system Huang et al., 2017 Narodytska and Kasiviswanathan, 2017.

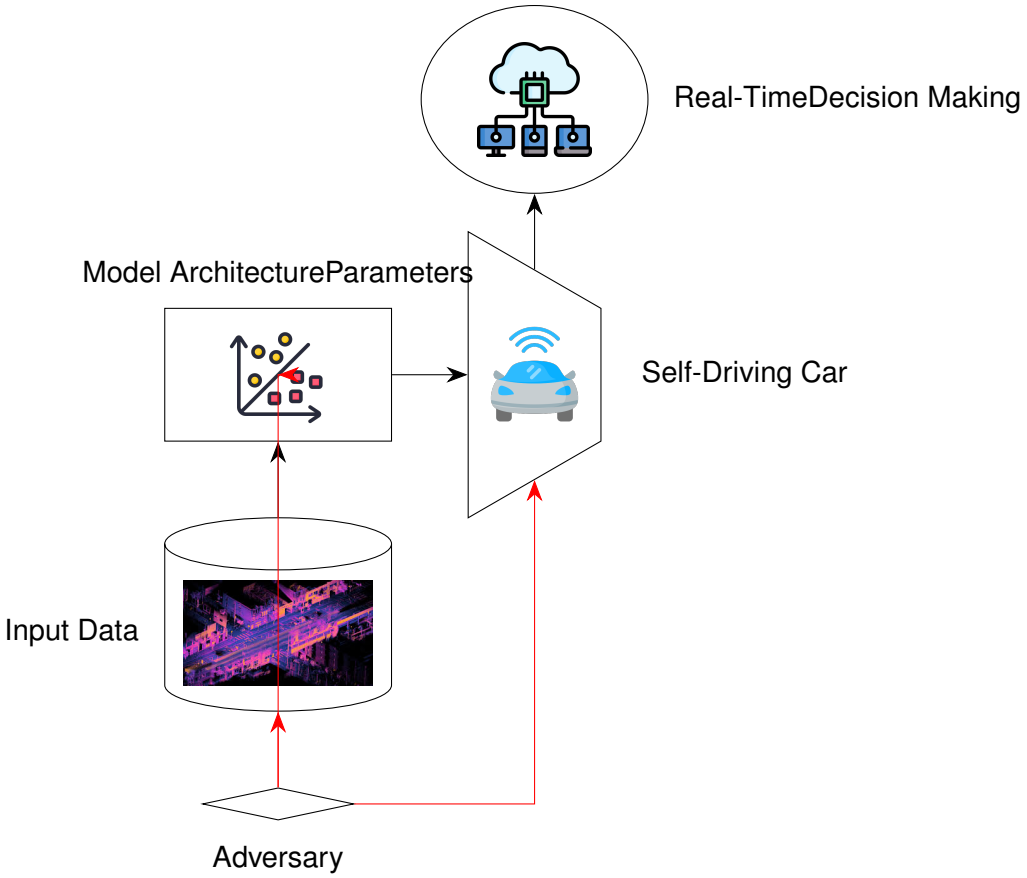

Figure 5: White-box adversarial setting in a self-driving car scenario.

In contrast to white-box attacks, black-box attacks operate under more limited access to the deployed model. In a black-box setting, the adversary does not have direct access to the model's internal details but can still interact with the model by sending inputs and observing the corresponding outputs.

Black-box attacks during deployment rely on the adversary's ability to probe the model and gather information about its behavior through carefully crafted queries. By sending different inputs to the model and analyzing the responses, the adversary can infer patterns, weaknesses, or inconsistencies that can be exploited to generate adversarial examples.

Figure 6 depicts a black-box attack scenario on a deployed model. In this setting, the adversary interacts with the model through an API or integration point, allowing them to send crafted inputs and observe the model's outputs. The adversary can systematically probe the model with various inputs, analyzing the responses to identify vulnerabilities or weaknesses in the model's decision-making process.

One common approach in black-box attacks is to exploit the transferability property of adversarial examples. Transferability refers to the phenomenon where adversarial examples generated for one model can also deceive other models with similar architectures or trained on similar data. By leveraging this property, adversaries can create adversarial examples using a surrogate model and then use those examples to attack the deployed model, even without direct access to its internals.

Black-box attacks during deployment can also target the model's integration with other systems or the input data pipeline. Adversaries may at-

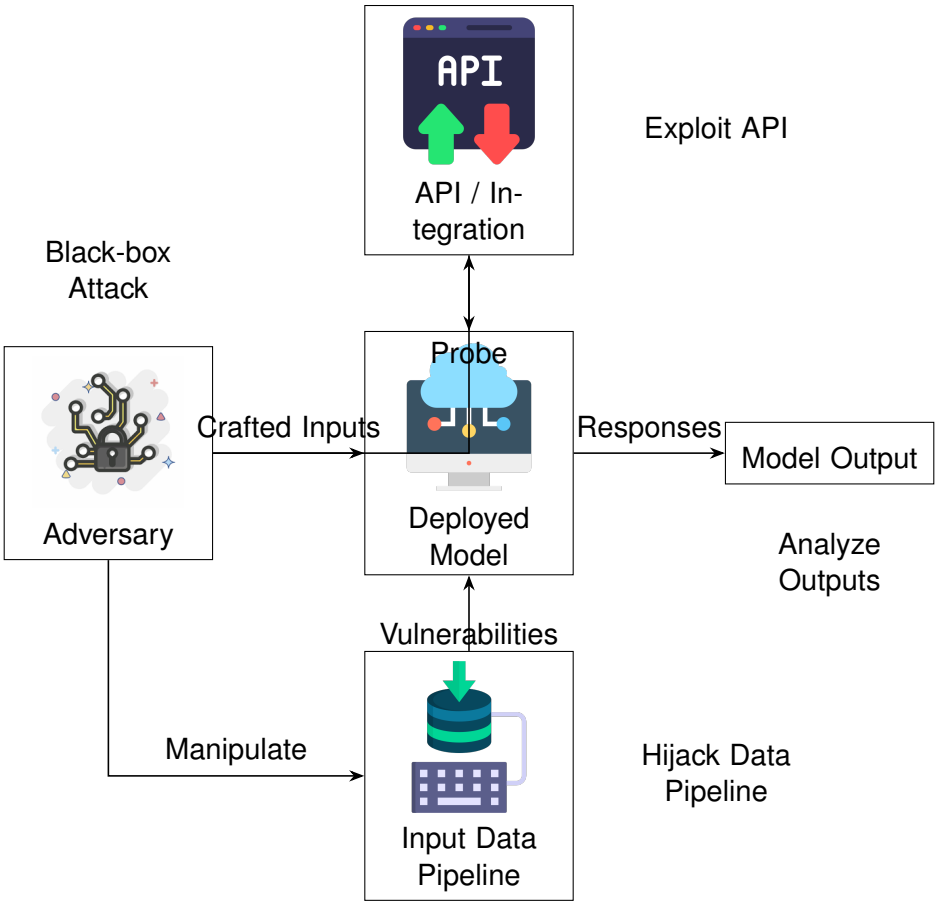

Figure 6: Black-box attack scenario on a deployed model with images.

tempt to manipulate the data being fed into the model, either by injecting malicious data points or by hijacking the communication channels between the model and its data sources. By corrupting or poisoning the input data, adversaries can indirectly influence the model's predictions and cause it to make erroneous decisions.

The consequences of black-box attacks during deployment can be significant, as they can undermine the reliability and trustworthiness of the deployed model. Even without direct access to the model's internals, adversaries can exploit vulnerabilities and weaknesses to manipulate the model's behavior and cause unintended outcomes. This can lead to financial losses, reputational damage, or even physical harm in critical applications.

Real-World Impact of Adversarial Attacks during Deployment: The impact of adversarial attacks during the deployment stage extends beyond the immediate consequences on the targeted system. These attacks can have far-reaching implications for individuals, organizations, and society as a whole.

In domains such as healthcare, finance, and criminal justice, machine learning models are increasingly being used to make important decisions that directly affect people's lives. Adversarial attacks on deployed models in these domains can lead to misdiagnoses, financial losses, or wrongful convictions. The consequences of such attacks can be devastating, causing harm to individuals and eroding trust in the use of machine learning systems.

Moreover, adversarial attacks during deployment can undermine the reliability and trustwor-

Technique	Definitions
Runtime Monitoring	Define a monitoring function $M(x)$ that assesses the input for anomalies or suspicious patterns. If $M(x)$ exceeds a predefined threshold τ, trigger an appropriate defense mechanism $D(x)$: $D(x) = \begin{cases} \text{Reject } x & \text{if } M(x) > \tau \\ \text{Accept } x & \text{otherwise} \end{cases}$
Anomaly Detection	Define an anomaly score $A(x) = -\log p(x)$ that measures the deviation of x from the expected distribution. Flag input x as anomalous if $A(x) > \tau$ and trigger further investigation or defense mechanisms.
Input Validation	Define a validation function $V(x)$ that checks the input x against predefined constraints and rules. If $V(x)$ fails, reject the input: $\text{Input Validation}(x) = \begin{cases} \text{Reject } x & \text{if } V(x) \text{ fails} \\ \text{Accept } x & \text{otherwise} \end{cases}$
Data Encryption	Let $E_k(x)$ be an encryption function with key k that encrypts the input x. Before feeding the input to the model, encrypt it: $x^{encrypted} = E_k(x)$. Decrypt the input within a secure environment before processing: $x = D_k(x^{encrypted})$, where D_k is the decryption function with key k.
Access Control	Define an access control policy P that specifies the permitted actions and privileges for different entities. Enforce the policy during deployment to limit unauthorized access and manipulation of the model and its inputs.
Security Audits and Penetration Testing	Conduct regular security audits to assess the system's vulnerabilities and risks. Perform penetration testing by simulating adversarial attacks to identify weaknesses and improve the system's defenses. Update the model, deployment practices, and defense mechanisms based on the findings of the audits and testing.

Table 3: Techniques for Defending Against Adversarial Attacks During Deployment

thiness of machine learning models in the eyes of the public. If deployed models are shown to be vulnerable to manipulation or deception, it can lead to a loss of confidence in the technology and hinder its adoption in critical applications. The public's trust in the decisions made by these models is crucial for their successful integration into various domains.

The deployment stage also presents challenges in terms of detecting and responding to adversarial attacks in real-time. Unlike the training or testing stages, where attacks can be identified and mitigated before the model is put into production, attacks during deployment require immediate detection and response mechanisms. The ability to quickly identify and neutralize adversarial inputs is essential to minimize the impact of the attack and maintain the system's integrity.

Furthermore, the evolving nature of adversarial attacks means that deployed models must be continuously monitored and updated to stay resilient against new and emerging threats. As adversaries adapt their techniques and discover new vulnerabilities, the deployed models must be regularly evaluated and strengthened to maintain their robustness and reliability.

Conclusion: Adversarial attacks during the deployment stage pose significant challenges and risks to the reliability, trustworthiness, and safety of machine learning models in real-world scenarios. Whether through white-box attacks that exploit detailed knowledge of the model's internals or black-box attacks that leverage the model's input-output behavior, adversaries can manipulate the model's decisions and cause unintended consequences.

The impact of these attacks extends beyond the immediate system, affecting individuals, organizations, and society as a whole. The consequences can be severe, ranging from financial losses and reputational damage to physical harm and loss of public trust in machine learning technology.

As machine learning models become increasingly integrated into critical decision-making processes, it is crucial to prioritize research and development efforts in adversarial robustness and resilience. Developing effective detection and response mechanisms, as well as regularly updating and strengthening deployed models, are essential steps in mitigating the risks posed by adversarial attacks.

The deployment stage represents a critical frontier in the battle against adversarial attacks, and it requires ongoing vigilance, collaboration, and innovation from researchers, practitioners, and stakeholders across various domains. By proactively addressing the challenges and risks associated with adversarial attacks during deployment, we can work towards building more secure, reliable, and trustworthy machine learning systems that can withstand the evolving landscape of adversarial threats.

Defending against adversarial attacks during the deployment stage requires a multi-faceted approach. Runtime monitoring is crucial to detect anomalous or suspicious inputs in real-time and trigger appropriate defense mechanisms. Let x be an input to the deployed model f_θ and define a monitoring function $M(x)$ that assesses the input for anomalies or suspicious patterns. If $M(x)$ exceeds a predefined threshold τ, trigger an appropriate defense mechanism $D(x)$: $D(x) = \begin{cases} \text{Reject } x & \text{if } M(x) > \tau \\ \text{Accept } x & \text{otherwise} \end{cases}$.

Anomaly detection techniques can be employed to identify inputs that deviate significantly from the expected distribution and flag them for further investigation. Let $\mathcal{X}$ be the set of expected inputs and $p(x)$ be the probability distribution of $x \in \mathcal{X}$. Define an anomaly score $A(x) = -\log p(x)$ that measures the deviation of x from the expected distribution. Set a threshold τ based on the desired false positive rate and flag input x as anomalous if $A(x) > \tau$, triggering further investigation or defense mechanisms.

Secure deployment practices, such as input validation, data encryption, and access control, can help minimize the attack surface and limit the adversary's ability to manipulate the model or its inputs. For input validation, define a validation function $V(x)$ that checks the input x against predefined constraints and rules, rejecting the input if $V(x)$ fails. For data encryption, let $E_k(x)$ be an encryption function with key k that encrypts the input x before feeding it to the model, and decrypt the input within a secure environment before processing: $x = D_k(x^{encrypted})$, where D_k is the decryption function with key k. For access control, define an access control policy P that specifies the permitted actions and privileges for different entities and enforce the policy during deployment to limit unauthorized access and manipulation of the model and its inputs.

Regular security audits and penetration testing can proactively identify and address vulnerabilities in the deployed system. Conduct regular security audits to assess the system's vulnerabilities and risks, and perform penetration testing by simulating adversarial attacks to identify weaknesses and improve the system's defenses. Update the model, deployment practices, and defense mecha-

nisms based on the findings of the audits and testing. The overall defense strategy can be formalized as: $\text{Defense}(x) = \begin{cases} \text{Reject } x & \text{if } M(x) > \tau \text{ or } V(x) \text{ fails} \\ f_\theta(D_k(x^{encrypted})) & \text{otherwise} \end{cases}$, where $\text{Defense}(x)$ represents the end-to-end defense mechanism applied to the input x during deployment, considering runtime monitoring, anomaly detection, input validation, and data encryption.

5. Conclusions

Adversarial examples represent a significant challenge in the field of machine learning, exposing a concerning vulnerability in these powerful systems. The ability of carefully crafted perturbations to deceive machine learning classifiers raises questions about the reliability and trustworthiness of these models in real-world applications. Adversarial capabilities can vary significantly across the different stages of the machine learning pipeline, which include training, testing, and deployment. Each stage presents unique challenges and opportunities for adversaries to manipulate or exploit the model. During the training stage, adversarial capabilities primarily revolve around the adversary's ability to manipulate or poison the training data used to build the machine learning model. In a white-box setting, the adversary has full access to the training data and can directly modify, add, or remove examples from the dataset. This level of access allows the adversary to introduce carefully crafted malicious examples that can mislead the model during training. By injecting specifically designed examples, the adversary can manipulate the model to learn incorrect patterns, biases, or behaviors that align with their malicious intent. For instance, an adversary could add a small perturbation to an image in the training set, causing the model to misclassify it as a different object. Repeated exposure to such manipulated examples during training can lead the model to internalize the adversary's desired behavior. In a black-box setting, the adversary may not have direct access to the training data but can still attempt to influence the training process indirectly. They can try to manipulate the data collection process by injecting malicious data points into the pipeline or exploiting weaknesses in the data preprocessing steps. For example, if the training data is collected from user-generated content, an adversary could flood the data collection system with manipulated or misleading examples. Even without direct access to the training data, the adversary can still impact the model's learning process by polluting the input data stream. Defending against adversarial attacks during the training stage involves several techniques. In data sanitization, the training data is carefully examined and filtered to remove any suspicious or potentially malicious examples. Anomaly detection methods can be employed to identify and flag unusual patterns or outliers in the training data. Additionally, robust training methods, such as adversarial training or training with noisy examples, can help the model become more resilient to adversarial perturbations. Explicitly incorporating adversarial examples into the training process helps the model learn to correctly classify them, thereby reducing its vulnerability to such attacks.

Testing Stage: Adversarial capabilities during the testing stage focus on the adversary's ability to craft adversarial examples that can deceive the trained model and cause it to make incorrect predictions or classifications. In a white-box setting, the adversary has complete knowledge of the model's architecture, parameters, and even access to the test data. Armed with this information, the adversary can analyze the model's internals and identify specific vulnerabilities or weaknesses. They can then generate highly targeted adversarial examples that exploit these vulnerabilities. For instance, by carefully perturbing input features or adding imperceptible noise to the test examples, the adversary can manipulate the model's output and force it to make incorrect predictions. White-box adversarial examples are dangerous because they are tailored to the specific model and can be highly effective in fooling it. In a black-box setting, the adversary does not have access

to the model's internal details but can still attempt to create adversarial examples by querying the model with crafted inputs and observing the outputs. The adversary can use techniques like transferability, where adversarial examples generated for one model can also fool other models with similar architectures. The adversary can create adversarial examples without direct access to the target model by exploiting the transferability property. They can train a surrogate model with a similar architecture and generate adversarial examples for it, which are then used to attack the target model. Defending against adversarial attacks during the testing stage involves various techniques. One effective approach is adversarial training, where the model is explicitly trained on adversarial examples alongside the original clean examples. Exposing the model to adversarial examples during training helps it learn to recognize and correctly classify them, thus making it more robust to such attacks. Ensemble methods, where multiple models are combined to make predictions, can also improve robustness by leveraging the collective decision-making of different models. Randomized defenses, such as adding random noise or applying random transformations to the input data, can make it harder for adversaries to craft effective adversarial examples. Deployment Stage: Adversarial capabilities during the deployment stage involve the adversary's ability to exploit the model in real-world scenarios, where the model is actively being used to make decisions or predictions. In a white-box setting, the adversary may have access to the deployed model's architecture, parameters, and even the input data being fed into the model. With this level of access, the adversary can analyze the model's behavior and identify potential vulnerabilities. They can craft targeted attacks that manipulate the model's output in real-time, potentially causing harm or compromising the system's integrity. For example, in a self-driving car scenario, an adversary with white-box access could manipulate the input data to mislead the model and cause the car to make dangerous decisions. In a black-box setting, the adversary does not have direct access to the model's internals but can still attempt to fool the deployed model by sending carefully crafted inputs and observing the model's responses. The adversary can probe the model with different inputs and analyze the outputs to infer patterns or weaknesses. They can then exploit these weaknesses by crafting adversarial examples that are likely to deceive the model in real-world scenarios. Additionally, the adversary can try to exploit vulnerabilities in the model's integration with other systems or APIs, such as manipulating the input data pipeline or hijacking the communication channels. Runtime monitoring is used to detect anomalous or suspicious inputs in real-time and trigger appropriate defense mechanisms. Anomaly detection techniques can also be employed to identify inputs that deviate significantly from the expected distribution and flag them for further investigation. Secure deployment practices, such as input validation, data encryption, and access control, can help minimize the attack surface and limit the adversary's ability to manipulate the model or its inputs. Regular security audits and penetration testing can proactively identify and address vulnerabilities in the deployed system.

The boundaries between these stages can sometimes be blurry, and adversarial capabilities can span across multiple stages. An adversary may use knowledge gained during the training stage to craft more effective attacks during testing or deployment. An all-inclusive defense strategy should consider adversarial capabilities across all stages of the machine learning pipeline. Addressing adversarial capabilities requires an all-inclusive approach that incorporates defense mechanisms at each stage. This includes robust training techniques to make the model resilient to adversarial perturbations, thorough testing and evaluation to identify and mitigate vulnerabilities, and secure deployment practices to protect the model in real-world scenarios. Monitoring, regular updates, and adaptive defense mechanisms are essential to keep pace with adversarial threats and maintain the model's robustness over time.

References

Bradshaw, J., Matthews, A. G. d. G., & Ghahramani, Z. (2017). Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*.

Cai, Q.-Z., Du, M., Liu, C., & Song, D. (2018). Curriculum adversarial training. *arXiv preprint arXiv:1805.04807*.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., & Song, L. (2018). Adversarial attack on graph structured data. *International conference on machine learning*, 1115–1124.

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

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*.

Ilyas, A., Engstrom, L., Athalye, A., & Lin, J. (2018). Black-box adversarial attacks with limited queries and information. *International conference on machine learning*, 2137–2146.

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

Liu, X., Yang, H., Liu, Z., Song, L., Li, H., & Chen, Y. (2018). Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*.

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

Narodytska, N., & Kasiviswanathan, S. P. (2017). Simple black-box adversarial attacks on deep neural networks. *CVPR Workshops*, *2*, 2.

Pang, T., Du, C., Dong, Y., & Zhu, J. (2018). Towards robust detection of adversarial examples. *Advances in neural information processing systems*, *31*.

Song, C., He, K., Wang, L., & Hopcroft, J. E. (2018). Improving the generalization of adversarial training with domain adaptation. *arXiv preprint arXiv:1810.00740*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.