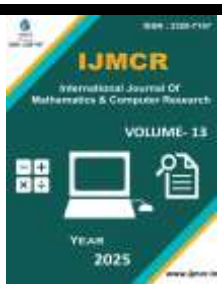
International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 13 Issue 07 July 2025, Page no. –5417-5429

Index Copernicus ICV: 57.55, Impact Factor: 8.615

DOI: 10.47191/ijmcr/v13i7.11



Adversarial Attacks and Vulnerabilities of AI in a Decision-Making System: Issues and Countermeasures

Augustin NYEMBO MPAMPI[1], Richard ILUNGA KALANDA[2], Pélagie MPEMBA MUKONKOLE[3], Jean Marie IBANGA MBAYO[4]

[1]Graduate in Computer Science and Independent Researcher

[2,3,4]Assistant at Notre Dame de Lomami University, Faculty of Computer Science

ARTICLE INFO

Published Online:
11 July 2025

Corresponding Author:
Augustin NYEMBO MPAMPI

ABSTRACT

Adversarial attacks exploit vulnerabilities in artificial intelligence models to alter their decisions, posing a major threat to critical domains such as cybersecurity, finance, and autonomous vehicles. Although various defense strategies, such as adversarial learning and anomaly detection, have been developed, they remain limited by their lack of generalization and their negative impact on model performance. The constant evolution of attack techniques renders these approaches insufficient, requiring an overhaul of protection strategies to ensure more resilient and secure AI.

To counter these threats, it is essential to design AI models that are more resilient to disruptions by integrating self-correction and adaptive learning mechanisms. Furthermore, securing deployment environments, encrypting models, and auditing queries are essential measures to strengthen their protection. Finally, close collaboration between researchers, industry, and regulators is necessary to establish robust security standards and ensure the reliable and seamless adoption of AI in critical infrastructure.

KEYWORDS: Adversarial attacks, artificial intelligence, model security, machine learning, robustness, adversarial learning , explainable AI, cybersecurity, deep learning , anomaly detection, model encryption, autonomous vehicles, data manipulation, security standardization, system resilience.

INTRODUCTION

The rise of artificial intelligence in decision-making systems has significantly transformed key sectors such as finance, healthcare, and autonomous vehicles. However, this expansion comes with new vulnerabilities, including the exposure of machine learning models to adversarial attacks . These attacks exploit flaws in algorithms to manipulate the models' decisions, compromising their reliability and security.

adversarial learning to anomaly detection. However, these approaches suffer from major limitations, including a lack of generalization and a negative impact on model accuracy. Furthermore, the constant evolution of attacks requires continuous rethinking of protection solutions to avoid a permanent race between defenders and attackers.

This article first examines adversarial attacks and vulnerabilities in AI models, before exploring protection methods and their limitations. It then proposes perspectives for improvement, focusing on the design of more robust models, securing AI environments, the explainability of algorithmic decisions, and the need for collaboration between researchers and industry to anticipate future threats.

1. ADVERSARY ATTACKS IN AI

1.1. Definition and principles

Adversarial attacks in artificial intelligence involve intentionally manipulating the inputs of a machine learning model to distort its decisions (Goodfellow et al., 2015). These attacks exploit mathematical vulnerabilities and algorithmic limitations of models, particularly deep neural networks, which are often sensitive to small variations in the data.

Adversarial attacks are a growing threat, particularly in facial recognition, cybersecurity, and autonomous vehicle systems. They can be classified based on their access to the model and their end goal.

1.2 . Typology of Adversarial Attacks

a) Mode of access to the model

- *White-box attacks* : The attacker knows the model's architecture, weights, and algorithm. They can then calculate gradients to generate targeted perturbations.
- *Black-box attacks* : The attacker has no knowledge of the model, but can interact with it and use exploration methods to understand its behavior.

b) Objective of the attack

- *Targeted attack* : The attacker forces the model to classify a specific input into an incorrect category.
- *Untargeted attack* : The attacker only seeks to destabilize the classification without aiming for a specific result.

Adversarial attack techniques and demonstrations

We illustrate two major types of attacks with real, executable codes.

1.3.1. FGSM (Fast Gradient Sign Method) attack

Principle

Developed by Goodfellow et al. (2015), FGSM consists of adding a minimal perturbation in the gradient direction of the loss function to maximize the model error.

$$x_{adv} = x + \in . sign(\nabla_x J(x, y))$$

Or :

- x is the original image,
- $\in$ is the intensity of the attack,
- $J(x, y))$ is the loss function of the model,
- $\nabla_x J(x, y)$ is the gradient of the loss with respect to the image.

Implementation in Python

We apply FGSM on MobileNetV2 , an image classification model.

```
import tensorflow ace tf
import numpy ace np
import matplotlib.pyplot ace plt
from keras.applications .mobilenet_v2 import MobileNetV2, preprocess_input , decode_predictions
from keras.utils import load_img , img_to_array

# Load the pre-trained MobileNetV2 model
model = MobileNetV2( weights = " imagenet ")

# Load and preprocess an image
img_path = "elephant.jpg"
try:
   img = load_img ( img_path , target_size =(224, 224) )
except FileNotFoundError :
   print ( f "Image not found: { img_path }")
   exit( 1)
img_array = img_to_array ( img )
img_array = np.expand _dims ( img_array , axis=0)
img_array = preprocess_input ( img_array )

# Original prediction
preds = model.predict ( img_array )
print ( "Predicted class before attack:", decode_predictions ( preds , top=1)[0])

# Generation of the FGSM attack
epsilon = 0.02
img_tensor = tf.convert _to_tensor ( img_array , dtype =tf.float32)
with tf.GradientTape () as tape:
   tape.watch ( img_tensor )
prediction = model( img_tensor )
   # Correction: loss should be calculated with from_logits = True
   label = tf.constant ([ np.argmax (preds)], dtype =tf.int32)
loss = tf.keras.losses.sparse_categorical_crossentropy (label ,
```

```
    prediction , from_logits = True )

# Calculating the gradient and adding a perturbation
gradient = tape.gradient (loss, img_tensor )
disturbance = epsilon * tf.sign (gradient)
img_adv = img_tensor + disturbance
img_adv = tf.clip _by_value ( img_adv , -1, 1)

# Prediction after attack
preds_adv = model.predict ( img_adv )
print ( "Predicted class after attack:", decode_predictions ( preds_adv , top=1)[0])

# Displaying images
plt.figure ( figsize =(10, 5))
plt.subplot (1, 2, 1)
# Fix: reverse normalization [-1,1] -> [0,1]
plt.imshow ( np.clip (( img_array [0]+1)/2, 0, 1))
plt.title (" Original image ")
plt.axis ('off')
plt.subplot (1, 2, 2)
plt.imshow ( np.clip (( img_adv.numpy ()[0]+1)/2, 0, 1))
plt.title ("Image after FGSM attack")
plt.axis ('off')
plt.show ()
```

This code implements a *Fast Gradient Sign Method (FGSM) attack* on the pre-trained *MobileNetV2 model* to trick its image classification.

Input image

This code starts by loading and preprocessing the image by resizing it to *224x224 pixels* and normalizing its values to match the model's expectations. After an initial prediction, it generates an adversarial perturbation by calculating the *gradient of the loss* with respect to the input image, then adding a small controlled variation to it. ($\in = \mathbf{0.02}$) in the direction that maximizes the model's error. This subtle manipulation alters the image in ways that are imperceptible to the human eye but *is enough to fool MobileNetV2* , resulting in misclassification after the attack. Finally, the

script displays the original and modified images, allowing the effect of the attack on the model's perception to be visualized.

Resized image

Image modified after adversarial disturbance

1.3.2. Data Poisoning Attack

Principle

The attack poisons training data by injecting malicious samples to bias a model (Biggio et al., 2018).

Implementation in Python

We modify the labels of a dataset (MNIST) and train an SVM on this corrupted data.

```
import numpy ace np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from tensorflow.keras .datasets import mnist

# Load MNIST
( x_train , y_train ), ( x_test , y_test ) = mnist.load_data ( )

# Normalize and flatten images
x_train = x_train.reshape ( -1.28 * 28 ) /255.0
x_test = x_test.reshape ( -1.28 * 28 ) /255.0

# Poisoning: Change 10% of labels
poison _percent = 0.1
num _poison = int ( len ( y_train ) * poison_percent )
np.random .seed ( 42 )
poison _indices = np.random.choice ( len ( y_train ), num_poison , replace=False)
y_train_poisoned = y_train.copy ( )
y _train_poisoned [ poison_indices ] = 9

# Training an SVM with corrupted data
model = SVC( kernel= "linear" )
model.fit ( x_train , y_train_poisoned )

# Assessment
accuracy = model.score ( x_test , y_test )
print ( "Accuracy after poisoning attack:" , accuracy )
```

This code implements a data poisoning attack on the MNIST dataset, used for handwritten digit recognition. It starts by loading MNIST and normalizing the images by flattening them into vectors of size **28×28 pixels** and dividing them by 255 **so** that their values are between **0 and 1** . Next, it introduces data corruption by randomly modifying 10% of the training labels, forcing these examples to be classified as **the digit "9"** , which biases the model's training. A Support Vector Machine (**SVM**) with a linear kernel is then trained on this corrupted dataset. Finally, the model's accuracy is evaluated on unmodified test data, revealing the impact of the attack on its performance, thus demonstrating how label manipulation during training can compromise a classification model.

1.4. Impact in a Real Case

Adversarial attacks have major implications for cybersecurity and autonomous systems. A concrete example was observed in self-driving cars. In 2019, researchers managed to fool an on-board vision system by slightly modifying a road sign: a simple sticker added to a STOP sign was enough to make it recognize it as a SPEED LIMIT sign, thus threatening the safety of passengers and pedestrians (Eykholt et al., 2018). This type of attack demonstrates the fragility of AI models in real-life conditions and the importance of strengthening their security.

Adversarial attacks pose a critical threat to AI models used in decision-making systems. The FGSM attack illustrates how a simple perturbation can fool a classification model, while the data poisoning attack demonstrates how changing training labels can bias a model over the long term. In real-world settings, these vulnerabilities pose a threat to cybersecurity systems, autonomous vehicles, and facial recognition devices. In the following section, we will explore countermeasures and solutions to strengthen the robustness of AI models against these attacks.

2. VULNERABILITIES OF AI SYSTEMS IN DECISION-MAKING

The integration of artificial intelligence (AI) into decision-making systems, particularly in finance, cybersecurity, healthcare, and autonomous vehicles, raises challenges related to reliability and security. Adversarial attacks demonstrate that these models are not infallible and that they possess structural and algorithmic vulnerabilities. These flaws can be exploited to alter predictions, bias decisions, or compromise system integrity. This section explores the main vulnerabilities of AI models, their origins, and their consequences.

2.2. Sensitivity to Adverse Disturbances

Deep models Learning , including convolutional neural networks (CNNs) and transformers used in natural language processing (NLP), are susceptible to near-invisible perturbations. A small, imperceptible change in an image, text, or audio signal can be enough to fool a model and distort a critical decision.

The following example shows how a CNN neural network can be fooled by an **FGSM attack** on an image:

```
import tensorflow ace tf
import numpy ace np
import matplotlib.pyplot ace plt
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input , decode_predictions
from tensorflow.keras.preprocessing import picture

# Load pre-trained MobileNetV2
model = MobileNetV2( weights = " imagenet " )

# Load an image and prepare it
img_path = "elephant.jpg"
img = image.load _img ( img_path , target_size = ( 224 , 224 ))
img_array = image.img_to_array ( img )
img_array = np.expand _dims ( img_array , axis= 0 )
img_array = preprocess_input ( img_array )

# Original prediction
preds = model.predict ( img_array )
print ( "Predicted class before attack:" , decode_predictions ( preds , top=1)[ 0 ])

# Generation of the FGSM attack
epsilon = 0.02
img_tensor = tf.convert _to_tensor ( img_array , dtype =tf.float32)
```

```
with tf.GradientTape () as tape:
    tape.watch ( img_tensor )
prediction = model( img_tensor )
loss = tf.keras .losses.sparse_categorical_crossentropy([np.argmax(preds)], prediction)

# Calculating the gradient and adding a perturbation
gradient = tape.gradient (loss, img_tensor )
disturbance = epsilon * tf.sign (gradient)
img_adv = img_tensor + disturbance
img_adv = tf.clip _by_value ( img_adv , - 1 , 1 )

# Prediction after attack
preds_adv = model.predict ( img_adv )
print ( "Predicted class after attack:" , decode_predictions ( preds_adv , top=1)[ 0 ])
```

This code applies a FGSM attack to an image and slightly modifies its pixels to trick MobileNetV2 into incorrectly classifying an object. Such a vulnerability can be exploited to thwart camera surveillance systems or manipulate facial recognition.

2.3. Exploitation of Data Bias

AI relies on training data, making it vulnerable to biases in that data. Biases can be unintentional (due to an unrepresentative dataset) or intentionally injected by an attacker.

Data poisoning involves altering training samples to bias the model and distort future decisions.

The following example illustrates how to modify labels in a dataset :

```
import numpy ace np
import matplotlib.pyplot ace plt
from sklearn.svm import SVC
from tensorflow.keras .datasets import mnist

# Load MNIST
( x_train , y_train ), ( x_test , y_test ) = mnist.load_data ( )

# Normalization and formatting of images
x_train = x_train.reshape ( -1.28 * 28 ) /255.0
x_test = x_test.reshape ( -1.28 * 28 ) /255.0

# 10% data poisoning
poison _percent = 0.1
num _poison = int ( len ( y_train ) * poison_percent )
np.random .seed ( 42 )
poison _indices = np.random.choice ( len ( y_train ), num_poison , replace=False)
y_train_poisoned = y_train.copy ( )
y _train_poisoned [ poison_indices ] = 9

# Training an SVM with poisoned data
model = SVC(kernel= " linear " )
model.fit ( x_train , y_train_poisoned )

# Accuracy Assessment
accuracy = model.score ( x_test , y_test )
print ( "Accuracy after data poisoning:" , accuracy )
```

This script deliberately modifies the labels of 10% of the MNIST data, forcing it to be classified as the number "9," thereby distorting the model's training. Such an attack could be used to circumvent a fraud detection system or bias a credit scoring model.

2.4. Excessive Reliance on Optimization Functions

AI models optimize cost functions using methods such as Stochastic Gradient Descent (SGD). These algorithms are vulnerable to attacks based on:

- Their sensitivity to gradients (used by FGSM, PGD, and DeepFool).
- Manipulating model weights through techniques such as gradient poisoning.

Example of instability due to optimization : A slight change in the weights can induce a drastic change in the predictions:

```
import tensorflow ace tf

# Creating a simple model
model = tf.keras.Sequential ( [
   tf.keras.layers.Dense ( 10 , activation= ' softmax ' , input_shape =( 5 ,))
])

# Weight initials
initial_weights = model.get_weights ( )

# Arbitrary modification of weights
modified_weights = [ w + np.random.normal ( 0,0.1 , w.shape ) for w in initial_weights ]
model.set_weights ( modified_weights )
```

This simple change in weight, although minimal, can completely alter the model's predictions, illustrating a key vulnerability of optimization algorithms.

2.5 . Lack of Verification and Security in Real Environment

AI-based decision-making systems are rarely monitored in production, making them vulnerable to black-box attacks and malicious testing. For example, an AI-based chatbot may be exposed to text-based attacks if:

- He does not filter inputs .
- He does not use self-correction mechanisms .
- He is based solely on probabilistic predictions .

Example of NLP manipulation

```
from transformers import pipeline

nlp = pipeline( "sentiment- analysis " )
print( nlp ("I love this product!")) # Normal
# Confusion attack
print( nlp ("I love this product but hate it too"))
```

An attacker can construct ambiguous sentences that distort sentiment analysis.

2.6. Real Impact: The Tesla Case and Physical Attacks

A real-life case of AI vulnerability was observed in 2019 when researchers proved that a minimal physical modification to a road sign could fool a Tesla autonomous vehicle. By adding stickers to a STOP sign, they induced a misclassification, making the vehicle believe it was a SPEED LIMIT sign. This case illustrates that even advanced AI can be fooled by well-designed attacks.

The vulnerabilities of AI systems in decision-making are multiple and critical. They stem from their sensitivity to disruptions, their reliance on data, adversarial attacks , and the lack of monitoring in production. These flaws pose a major risk to security, finance, and automation, thus requiring robust defense mechanisms that we will explore in the next section.

3. DEFENSE STRATEGIES AGAINST ADVERSARY ATTACKS

Faced with *adversarial attacks* , which exploit the vulnerabilities of artificial intelligence (AI) systems to induce classification or decision errors, several defense strategies have been developed. These strategies aim to *strengthen the robustness of models* by reducing their sensitivity to adversarial perturbations , by detecting attacks or by preventing their execution. We will explore the main defense

approaches against these attacks, illustrated by concrete implementations.

3.2. Adversarial Training Principle

Adversarial learning involves training a model with adversarial examples so that it learns to recognize and process them correctly. This method, introduced by Goodfellow et al. (2015), is one of the most effective for improving model robustness.

Implementation in Python

adversarial examples generated with FGSM.

```
import tensorflow ace tf
import numpy ace np
from tensorflow.keras .datasets import mnist
from tensorflow.keras .models import Sequential
from tensorflow.keras .layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam

# Load MNIST data
( x_train , y_train ), ( x_test , y_test ) = mnist.load_data ( )
x_train , x_test = x_train / 255.0 , x_test / 255.0

# Building the simple CNN model
model = Sequential( [
   Flatten( input_shape =( 28 , 28 )),
   Dense( 128 , activation= ' relu ' ),
   Dense( 10 , activation= ' softmax ' )
])
model. compile (optimizer=Adam(), loss= ' sparse_categorical_crossentropy ' , metrics=[ 'accuracy' ])

adversarial examples with FGSM
def generate_adversarial_ example ( model, x, y, epsilon= 0.2 ):
   x_tensor = tf.convert _to_tensor ( x.reshape ( 1 , 28 , 28 ), dtype =tf.float32)
   with tf.GradientTape () as tape:
      tape.watch ( x_tensor )
prediction = model( x_tensor )
loss = tf.keras.losses.sparse_categorical_crossentropy ([y], prediction )
gradient = tape.gradient (loss, x_tensor )
disturbance = epsilon * tf.sign (gradient)
   return x_tensor + perturbation

# Creation of a mixed batch ( data normal + adverse )
x_train_adv = np.array ([ generate_adversarial_example (model, x, y). numpy ().reshape( 28 , 28 ) for x, y in zip (
x_train [: 1000 ], y_train [: 1000 ])])
x _train_mixed = np.concatenate ([ x_train , x_train_adv ])
y _train_mixed = np.concatenate ([ y_train , y_train [: 1000 ]])

# Training with augmented data
model.fit ( x_train_mixed , y_train_mixed , epochs = 5 , batch_size = 32 , validation_data = ( x_test , y_test ))
```

This code:

1. Loads the MNIST dataset and trains a dense neural network.
2. adversarial examples using FGSM.
3. adversarial data to train a more robust model.

3.3. Attack Detection and Filtering Principle

Adversarial attacks can be detected through statistical analysis of inputs or through the use of specialized neural networks capable of identifying anomalies in the data.

Python Implementation: Autoencoder Detection

An autoencoder is a neural network trained to reconstruct an input data. If an input is adversarial , the autoencoder will have difficulty reconstructing it correctly, thus revealing a potential attack.

```
from tensorflow.keras .models import Model
from tensorflow.keras .layers import Input, Dense

# Autoencoder definition
input_layer = Input(shape =( 28 * 28 ,))
encoded = Dense( 64 , activation= ' relu ' )( input_layer )
decoded = Dense( 28 * 28 , activation= 'sigmoid' )(encoded)

autoencoder = Model( input_layer , decoded)
autoencoder. compile (optimizer= ' adam ' , loss= ' binary_crossentropy ' )

# Training on MNIST (without adversarial attacks )
x_train_flattened = x_train.reshape ( -1.28 * 28 )
autoencoder.fit ( x_train_flattened , x_train_flattened , epochs= 5 , batch_size = 256 , shuffle= True )

# Detection of a suspicious image
def detect_adversarial_example ( autoencoder, x_input ) :
   x_reconstructed = autoencoder.predict ( x_input.reshape ( 1 , -1 ))
error = np.mean ( np.abs ( x_input - x_reconstructed ) )
   return error > 0.1 # Anomaly detection threshold

# Test on an example
print ( "Is the image suspicious?" , detect_adversarial_example ( autoencoder , x_test [ 0 ]))
```

An autoencoder is trained on normal images. An error threshold is set to detect adversarial images . If an image cannot be reconstructed correctly, it is considered suspicious.

3.4. Data Transformation Based Approaches Principle

adversarial attacks can be circumvented by applying transformations to the data before it is processed by the model:

- *Defense -GAN* : Uses a generative network to purify inputs.
- *Quantization and Rounding : Reduces* adverse disturbances by limiting pixel precision.

Example: Pixel Quantization

```
def quantize_image ( x, levels= 10 ) :
   return n.p. round (x * levels) / levels

x_test_quantized = quantize_image ( x_test )
preds = model.predict ( x_test_quantized )
```

The attack is weakened because **small disturbances** are rounded to fixed values, limiting their impact.

3.5 . Securing Models and Infrastructure

AI models must be protected against unauthorized access attempts and white-box attacks.

Security Measures

- *Isolating models in production* : Limiting access to neural network weights.
- *Homomorphic encryption* : Encrypt model weights to prevent their manipulation.
- *Query monitoring* : Detect suspicious queries (e.g. an attacker testing many inputs).

Example: Access Limitation with TensorFlow Serving

```
tensorflow_model_server -- rest_api_port =8501 -- model_name = secure_model -- enable_batching = false
```

TensorFlow server with restrictions on model access.

3.6. Real Impact and Case Study

Adversarial attacks and their countermeasures have direct implications for the security of AI systems:

- *Autonomous vehicles* : Tesla was vulnerable to an attack where stickers on a STOP sign induced a false classification, putting passenger safety at risk.
- *Facial recognition* : Some biometric systems have been fooled by special glasses that alter the perception of facial features.
- *Cybersecurity* : AI-based antiviruses can be bypassed by slightly modifying the source code of a malware.

These cases demonstrate the urgency of strengthening the security of AI models and adopting multi-level strategies to counter adversarial attacks .

Adversarial attacks pose a major risk to the reliability and security of AI systems. To defend against them, it is crucial to combine adversarial learning , attack detection, data transformation, and model security. As attackers develop more sophisticated techniques, research must continue to innovate to ensure robust and resilient AI models.

4. DISCUSSION AND PERSPECTIVES

4.1. Discussion

Adversarial attacks represent a **major challenge** in the use of artificial intelligence (AI) for decision-making. As discussed in previous sections, these attacks exploit mathematical, algorithmic, and systemic vulnerabilities to fool machine learning models. The effectiveness of attacks such as FGSM, PGD, and data poisoning demonstrates that even the most powerful AI models remain vulnerable to well-orchestrated disruptions.

adversarial learning , anomaly detection, and filtering techniques, can mitigate these risks, but no single solution guarantees absolute protection. Indeed, some countermeasures, such as adversarial training, improve the robustness of models but may reduce their accuracy on non-adversarial data . Similarly, autoencoders and detection methods can identify attacks, but at the cost of high computational cost and a possible increase in false positives.

Despite advances in protection against adversarial attacks , current defense strategies remain insufficient in the face of rapidly evolving threats. Three major limitations compromise the effectiveness of existing approaches: their lack of generalization, the inevitable trade-off between robustness and performance, and the constant adaptation of attackers to circumvent the countermeasures put in place.

The first limitation of defense approaches lies in their lack of generalization. Indeed, the majority of protection strategies are developed to counter a specific type of attack, but fail when faced with more sophisticated threats.

> ***For example ,*** adversarial learning , which involves training a model on adversarial examples generated by a specific attack such as FGSM (Fast Gradient Sign Method), allows to increase the robustness of the model against this particular attack.

However, it has been shown that more advanced attacks, such as the Carlini & Wagner (C&W) attack or the Projected Gradient Descent (PGD) attack, remain effective against models that have been previously hardened against FGSM.

This phenomenon is explained by the fact that artificial intelligence models, especially those based on deep learning, learn to recognize specific patterns of adversarial noise , but do not develop defense mechanisms that can be generalized to all attacks. Thus, an attacker informed of the defense strategy employed can design a slightly modified attack that bypasses the protections in place. Second, a trade-off between robustness and performance constitutes a major constraint for AI models subject to defenses against adversarial attacks .

When a model is trained to be more robust to perturbations, it often loses performance on unmodified data. This degradation is explained by the fact that increasing robustness involves changing the model's decision boundaries, which can impair its ability to classify normal examples well.

Thus, an overly robust model risks overgeneralizing and introducing classification errors on legitimate data, reducing its effectiveness in critical applications.

> ***For example*** , in the field of facial recognition, advanced defenses against adversarial attacks can result in false negatives, i.e., rejecting valid individuals, thus compromising user experience and technology adoption.

This trade-off is particularly problematic in real-time systems, such as those used for banking fraud detection or autonomous driving, where a decrease in accuracy can have serious consequences.

Finally, one of the most critical challenges is the constant evolution of adversarial attacks , making AI security a never-ending race between attackers and defenders. Each new defense method in turn leads to the development of more sophisticated adversarial strategies , creating a cycle of continuous improvement in attacks. Initially, adversarial attacks primarily targeted numerical models, exploiting mathematical flaws in deep neural networks to induce classification errors.

physical attacks , making the models even more vulnerable in real-life conditions.

> ***A concrete example*** is self-driving cars: initially, attacks involved digitally altering images of road signs, but researchers quickly demonstrated that slight physical changes were enough to fool vehicle sensors.

Simply placing stickers on a STOP sign can make it appear as a speed limit, which can have dramatic consequences for road safety. Similarly, strategically placed reflective objects on the road can destabilize onboard vision systems, causing an autonomous vehicle to make erroneous decisions.

These attacks demonstrate that AI models are not only vulnerable to digital modifications, but also to real-world disruptions, further complicating the defense strategies to be adopted.

adversarial defense approaches suffer from significant limitations that hinder their long-term effectiveness. Their lack of generalization prevents effective protection against a wide range of threats, while the trade-off between robustness and performance poses a difficult dilemma for AI system designers.

Furthermore, the rapid evolution of attacks, particularly with the shift from the digital to the physical world, shows that the security of models cannot be guaranteed absolutely. Faced with these challenges, it is imperative to explore new, more adaptive defense strategies that take into account both the effectiveness of models under normal conditions and their resistance to sophisticated attacks.

4.2 . Perspectives

The constant evolution of adversarial attacks and the limitations of current defense approaches require careful consideration of strategies to strengthen the robustness of artificial intelligence models. Several avenues of research and innovation can be considered to improve the resilience of systems against these threats. This involves not only optimizing the models themselves, but also securing their deployment environments, making them more explainable, and fostering collaboration between stakeholders in the field to better anticipate future threats.

4.2.1. Towards More Resilient AI Models

A first area of improvement is to design artificial intelligence models that are intrinsically more robust to adversarial attacks . This approach relies on modifying the architectures of neural networks to reduce their sensitivity to disturbances. Inspired by the biological mechanisms of the human brain, the integration of neuronal redundancy and self-correction capacity could allow models to absorb disturbances without altering their predictions. The idea would be to equip networks with internal self-repair mechanisms, capable of detecting and neutralizing adversarial inputs before they affect the decision-making process.

Another promising avenue is multitask learning, which involves training a model on multiple tasks simultaneously to improve its generalization and robustness to attacks. A model trained to recognize diverse data types and perform multiple classifications will be less vulnerable to targeted modifications to a single input category. This diversification of tasks reduces the chances that a specific disruption could affect the entire system.

Adding random noise during training is another technique for improving model robustness. This approach, which involves introducing controlled perturbations during the learning phase, forces the model to adapt to unforeseen variations and not rely solely on small-amplitude details in the input data. Thus, the model becomes more resilient to adversarial attacks that exploit these structural weaknesses. A particularly promising approach in this area is meta-learning, which allows machine learning models not only to learn from data, but also to learn to adapt to attacks. Using this technique, a model could adjust its classification strategies in real time when it detects potentially adversarial inputs .

4.2.2. Securing Deployment Models and Environments

The robustness of artificial intelligence systems depends not only on the characteristics of the models themselves, but also on the infrastructures and environments in which they are deployed. It is therefore essential to implement multi-layered safeguards to prevent any attempts to compromise AI models in production.

Model encryption is a key solution for protecting neural networks from exfiltration and malicious manipulation of their weights. Homomorphic encryption techniques make it possible to perform inferences on encrypted models without requiring their decryption, thus preventing any tampering attempts by an attacker.

Another key measure is runtime isolation, which limits access to production models to prevent attackers from conducting black-box attacks. By restricting entry points and enforcing strict validation mechanisms on queries sent to models, it becomes more difficult for an attacker to gain actionable insights into their operation.

Finally, continuous query auditing allows for real-time monitoring of AI model inputs to detect attack attempts early. Behavioral analytics and anomaly detection systems can identify suspicious inputs and block their execution before they impact decision-making. These measures are particularly critical in areas such as facial recognition, algorithmic finance, and cybersecurity, where model compromise can have serious consequences.

4.2.3. Explainable and Verifiable AI

One of the major challenges of adversarial attacks is that deep learning models operate as black boxes, making it difficult to explain the decisions made by the algorithm. This opacity makes it difficult for experts to identify the exact reasons for a classification error or a disruption introduced by an attack. Developing explainable and verifiable AI models could help strengthen their security and reliability.

Explainable AI would offer the ability to identify weaknesses in a neural network in advance, preventing an attacker from exploiting them to their advantage. Interpretability tools, such as SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model- Agnostic Explanations), allow us to better understand the decision-making process of the models and to anticipate potential vulnerabilities.

In critical systems, adding validation checks to AI decisions would provide an additional line of defense against adversarial attacks . An AI capable of providing a detailed justification for its decision could be verified by supervisory systems before an action is taken.

Explainable AI could also be integrated into self-defense mechanisms, where a model would learn to detect suspicious inputs and flag an anomaly when it identifies a potential attack. This approach would reduce reliance on static protection solutions and allow for more dynamic and reactive defense strategies.

4.2.4. Collaboration between Research and Industry

adversarial attacks constantly evolving, it is imperative to strengthen collaboration between researchers, industry, and regulators to implement effective defense strategies adapted to new threats. Currently, research on adversarial attacks is fragmented, making it difficult to implement standardized and comprehensive solutions.

A critical first step would be the development of public databases of adversarial attack examples . These databases would allow researchers and companies to test and improve their models against a wide range of known threats.

The establishment of AI-specific security standards would also be necessary to ensure that deployed artificial intelligence models meet minimum robustness criteria. Similar to ISO standards for cybersecurity, a regulatory framework dedicated to AI models would ensure a certain level of protection before they are put on the market.

Finally, the active involvement of regulatory authorities would be essential to regulate the use of AI in critical areas. The European Union and the United States have already begun implementing standards for securing AI models, but these initiatives must be expanded and adopted globally to be truly effective.

In conclusion, improving the security of artificial intelligence systems against adversarial attacks relies on several complementary axes. Designing more robust models, securing their execution environment, developing explainable AI, and implementing a rigorous regulatory framework are all essential avenues for ensuring the reliability of AI models. By integrating these different approaches, it becomes possible to design systems that are more resilient to emerging threats, thus strengthening confidence in the use of artificial intelligence in critical areas.

CONCLUSION

Adversarial attacks pose a growing threat to AI, compromising its integrity in critical applications. Deep learning models Learning models , due to their sensitivity to perturbations, are particularly vulnerable to adversarial manipulation . Although defense techniques have been developed, they remain insufficient, as they do not cover all forms of attacks and can negatively affect the performance of models on normal data.

To strengthen the robustness of AI systems, it is essential to design more resilient models that can better adapt to attacks by integrating self-correction and adaptive learning mechanisms. Furthermore, securing deployment environments and developing explainable AI are key areas for anticipating and detecting attacks before they compromise critical decisions.

Finally, combating adversarial attacks requires collaboration between researchers, businesses, and regulators to define appropriate security standards. Ensuring more transparent, robust, and secure AI will help ensure its confident integration into critical infrastructure and maximize its benefits for society.

REFERENCES

1. Biggio , B., & Roli , F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning . Pattern Recognition , 84, 317-331. https://doi.org/10.1016/j.patcog.2018.07.023
2. Carlini , N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks . IEEE Symposium on Security and Privacy (S&P) , 39-57. https://doi.org/10.1109/SP.2017.49
3. Eykholt , K., Evtimov , I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust physical-world attacks on deep learning models . Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , 1625-1634. https://doi.org/10.1109/CVPR.2018.00175
4. Goodfellow, I.J., Shlens , J., & Szegedy , C. (2015). Explaining and harnessing adversarial examples . International Conference on Learning Representations (ICLR) . https://arxiv.org/abs/1412.6572
5. Kurakin , A., Goodfellow, I., & Bengio , S. (2017). Adversarial examples in the physical world . International Conference on Learning Representations (ICLR) Workshop Track . https://arxiv.org/abs/1607.02533
6. Madry , A., Makelov , A., Schmidt, L., Tsipras, D., & Vladu , A. (2018). Towards deep learning models resistant to adversarial attacks . International Conference on Learning Representations (ICLR) . https://arxiv.org/abs/1706.06083

7. Moosavi-Dezfooli , SM, Fawzi, A., & Frossard, P. (2016). DeepFool : A simple and accurate method to fool deep neural networks . IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , 2574-2582. https://doi.org/10.1109/CVPR.2016.282
8. Papernot , N., McDaniel, P., Goodfellow, I., Jha, S., Celik , ZB, & Swami, A. (2017). Practical black-box attacks against machine learning . Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (AsiaCCS'17) , 506-519. https://doi.org/10.1145/3052973.3053009
9. Rosenberg, M., Shabtai, A., Rokach , L., & Elovici , Y. (2018). Adversarial machine learning attacks and defense methods in the cyber security domain . ACM Computing Surveys (CSUR) , 51(5), 1-36. https://doi.org/10.1145/3230630
10. Xu, W., Evans, D., & Qi, Y. (2020). Feature squeezing: Detecting adversarial examples in deep neural networks . Network and Distributed System Security Symposium (NDSS) . https://arxiv.org/abs/1704.01155