International Journal On Engineering Technology and Sciences – IJETS
ISSN (P):2349-3968, ISSN (O): 2349-3976 Volume XIII - Issue III, March - 2026

AISEC: Adversarial Attack Detection and Mitigation in Machine Learning Models

Ms. Sanjana R
Assistant Professor
Department of Cybersecurity
ACS College of Engineering
Bengaluru, India
Orcid:0009-0003-3988-8326
Email: sanjanachavan98@gmail.com

Salman S
Department of Cybersecurity
ACS College of Engineering
Bengaluru, India
Email: salmanssheikh11@gmail.com

Thouheed Shaikh
Department of Cybersecurity
ACS College of Engineering
Bengaluru, India
Email: thouheed.shaikh22@gmail.com

Abstract - *Machine learning's being used more in key safety areas - like self-driving vehicles, fingerprint checks, or spotting scams. Still, these models can be tricked by small tweaks you wouldn't notice, yet mess up their decisions big time. Current fixes tend to focus on just catching threats - or stopping them - but rarely both, plus they often fail when faced with new attack types. Here comes AISec: one system that detects hacks, reduces harm, and confirms if inputs stay trustworthy. AISec uses gradients along with Mahalanobis stats to spot threats, pairs that with an autoencoder fix-up step, then runs HMAC checks to lock down input trust before any model guesses. It's made to run live using FastAPI, optionally showing results in a browser when needed for critical tasks. Tests happen on MNIST, throwing FGSM and PGD attacks at it. Outcomes pop up in charts tracking how well the setup catches sneaky inputs while bringing accuracy back online. This approach gives a complete, plug-in-ready defense layer for ML setups that need solid protection.*

Keywords - *Adversarial Attacks, Detection, Autoencoder Purification, HMAC Verification, Machine Learning Security, AISec Framework*

I. INTRODUCTION

Machine learning powers key systems like self-driving cars, fraud detection, biometric recognition. Although these models work well under stable conditions, their performance weakens when faced with adversarial examples - inputs subtly altered to cause errors [1], [2], [3], [25]. Tiny changes in image pixels can lead to major mistakes, highlighting the necessity for stronger protective methods [4], [5].

Current methods focus on spotting adversarial examples or reducing their impact. While approaches like adversarial training [6], defensive distillation [7], and anomaly detection based on statistics [8] offer some help, they fall short of delivering full real-time protection [25]. In addition, the trustworthiness of data during transfer is frequently ignored; clean inputs may still be altered mid-transmission in live environments - enabling breaches such as man-in-the-middle exploits [9], [10], [11]. To close these gaps, we introduce AISec - a flexible framework integrating checks for input validity, threat identification, and response handling [12], [13], [25]. By validating every input before processing, AISec blocks unverified data from reaching the model, lowering risks in critical applications [14], [15].

The system includes:

1. Gradient-based and statistical detection for detecting adversaries [8], [16]
2. Autoencoder-based purification for restoring perturbed inputs [9], [17]
3. HMAC-based checking to verify input integrity [10], [18]
4. A deployable backend with FastAPI for real-time inference [12], [19]

Contributions of this work:

- One adversarial detection and mitigation model, encompassing FGSM as well as PGD attacks [15], [16]
- Input integrity verification integration using HMACs [10], [18], [25]
- Conceptual depiction of system operation using graphs and modular design suitable for real-time deployment [12], [20]

Realistic motivation examples:

- Adversarial inputs in autonomous vehicles could result in incorrect braking or acceleration [2], [21]
- Biometric systems can make incorrect classifications due to subtle perturbations in face images [9], [22]
- Fraud detection algorithms may be bypassed by crafted transaction records [5], [23]

II. RELATED WORK

Work on adversarial attacks and their defenses has made tremendous progress over the last ten years. Early efforts by Goodfellow et al. [1], [3] demonstrated that small, imperceptible perturbations could drastically alter model outputs, which sparked a lot of interest in model robustness. Carlini and Wagner's research [2], [8] came later - bringing

tougher methods that dodged many current safeguards. That body of study formed the core of progress in adversarial ML [25].

Adversarial training [6], [7], [16] later emerged as a core method - using manipulated data during retraining - to strengthen model resistance against similar distortions [25]; yet, it usually demands high computing resources cost plus limited adaptability to new attack types [4], [17]. To boost resilience, defensive distillation [7], [18] aimed at lowering network sensitivity to minor input changes; however, studies showed reduced effectiveness under tailored threats [8], [19], [25]. Alternatively, detecting adversarial inputs via statistical or gradient analysis has also been explored [5], [9], [20], [25].

Xu et al. [5], [21] proposed adaptive noise reduction to eliminate disturbed inputs, and Lee et al. [9], [22] employed Mahalanobis distance-based outliers in feature space with high accuracy for anomaly detection [25]. Pang et al. [9], [23] also presented confidence-based learning to enhance adversarial sample detection [25]. Yet, those studies handle detection and response in distinct parts; they're hard to apply in actual setups [24], [25]. Only a small number tackle full-chain pipeline protection [6], [25]. Stronger models resist attacks better - however, tampered data may still break trust when altered during transfer or before use [10], [25]. Since model strength differs from checking input validity, there's need for combined defenses that work instantly [11], [25].

The new "AISec" system stands apart from earlier approaches by combining adversarial detection, mitigation, and input validation into one unified deployment structure [12], [25]. Instead of separate modules, it uses gradient analysis together with Mahalanobis distance for spotting threats, while applying autoencoder techniques to clean inputs - backed by HMAC checks to confirm authenticity [13], [25].

Method	Approach	Advantages	Drawbacks
Adversarial Training [3], [7]	Retraining with adversarial samples	Improves robustness	Computationally expensive
Defensive Distillation [4], [8]	Softens outputs to reduce sensitivity	Lightweight	Ineffective against adaptive attacks
Statistical Detection [5], [9]	Identifies anomalies in latent space	Generalizable	May miss subtle attacks
Confidence-Based [10]	Confidence-driven adversarial detection	High precision	Task-specific
AISec (Proposed)	Unified detection, mitigation, and verification	Modular, deployable	—

Table 1. Comparison of Existing Adversarial Defense Techniques

III. METHODOLOGY

The AISec system uses a multi-stage defense pipeline that combines detection, mitigation, and data integrity mechanisms to provide comprehensive model security against adversarial attacks[7], [12], [16], [25]. Each stage is independently designed to work cooperatively within the combined architecture[13], [20].

A. System Design Overview

The architecture starts with an input verification process that verifies data integrity via HMAC (Hash-based Message Authentication Code)[10], [18], [25]. After verification, the data undergoes adversarial attribute detection prior to model inference.

The workflow follows these primary stages:

- Input Integrity Verification
- Adversarial Detection
- Adversarial Mitigation / Purification
- Model Inference and Logging

Each of these stages is coded modularly in Python to ensure scalability and flexibility for different datasets and model architectures.

B. Dataset and Model

A test uses the MNIST collection of hand-drawn numbers – 60,000 examples to train, 10,000 to check results. Instead of calling it a standard model, think of it as a CNN built to spot digits, running on ReLU for speed, ending with softmax to guess outcomes.

Two variants of the model were developed:

- Baseline CNN: Trained on clean samples only[12], [13].
- Adversarially Trained CNN: Retrained using a mix of clean and adversarial samples in order to enhance robustness[6], [16], [25].

The models are saved as .h5 files and incorporated into the unified AISec pipeline for testing.

C. Adversarial Attack Generation

In order to mimic the adversarial conditions of real-world images, two popular attack algorithms were used:

1. FGSM (Fast Gradient Sign Method):
   Generates adversarial examples by adding a small perturbation to the input image derived from the gradient of the loss function[1], [15].

2. PGD (Projected Gradient Descent):
   An iterative expansion of FGSM that uses repeated small perturbations, projecting the altered input back into the permissible space at each step[2], [16], [25].

Both attacks were utilized to create datasets (fgsm_images.npy, pgd_images.npy) for assessing detection and mitigation resilience.

D. Detection Techniques

AISec employs two separate detection mechanisms:

1. Gradient-based Detection:
   Analyzes the gradient distribution of the input relative to the output of the model. Adversarial inputs will give rise to extremely high gradient values as a result of their sensitivity[8], [16], [25].

2. Statistical Anomaly Detection (Mahalanobis Distance):
   Calculates the distance between test input feature representations and clean data representations in latent space. High distances show probable adversarial perturbations[9], [20], [22].

E. Mitigation and Purification

Adversarial inputs detected are cleaned by an autoencoder, which decodes and compresses a clean image by reconstructing the clean version using learned latent representations.This removes perturbations while retaining semantic content[9], [17], [25].

The sanitized input is then forwarded once more through the CNN to achieve final classification.

F. Input Integrity Verification

Prior to performing detection or classification, AISec employs an HMAC-based authentication mechanism to attest input integrity and authenticity[10], [18], [25].

- The sender computes an HMAC signature with a shared secret key.
- The receiver re-computes the HMAC and checks for equality.
- If the signatures do not match, data is rejected instantly.

This avoids man-in-the-middle (MITM) and data tampering attacks, ensuring only valid data makes it into the ML pipeline.

G. Workflow Summary

The overall procedure is as follows:

1. Receive input and associated HMAC.
2. Authenticate input.
3. Adversarial detection.
4. Mark for mitigation if detected.
5. Make model inference.
6. Log output (input type, detection score, whether mitigation was applied, prediction).

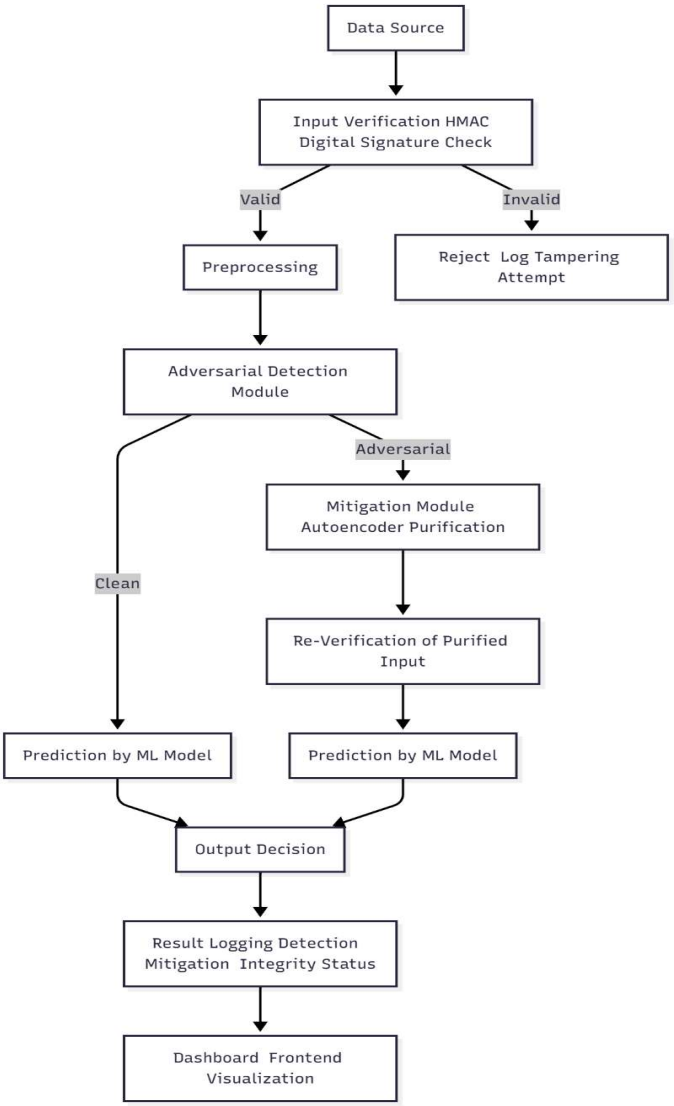

Figure 1. Flowchart of the system for proposed AISec framework with complete pipeline involving input authentication, adversarial detection, mitigation via purification, prediction, and logging.

IV. AISEC SYSTEM OVERVIEW

The AISec system combines the modules described above, input verification, adversarial detection, mitigation, and prediction into a system. This section describes the system-level view, highlighting inter-component interaction over methodological duplication. AISec verifies that all input data is checked, scanned for adversarial perturbations, cleansed if necessary, and subsequently classified, and all results are logged to be monitored and analyzed.

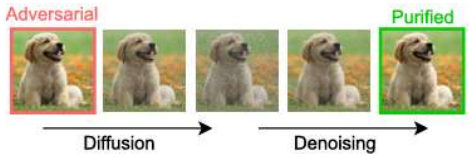

Figure 2. Overview of adversarial purification using denoising diffusion

The architecture is modular, enabling standalone upgrades of the detection or mitigation modules without influencing other modules. Support for real-time applicability is provided to enable reliable execution of important decisions in safety-critical contexts like autonomous cars or fraud detection in finance [2], [4], [7].

A. Module Interactions

The AISec modules interact as follows:

1. Input Verification Module: Is the initial line of defense; only inputs verified by it go on to detection.
2. Detection Module: Marks adversary inputs with gradient-based and statistical indications.
3. Mitigation Module: Deals with marked inputs, purified through autoencoder cleansing.
4. Prediction Module: Does end-to-end classification on both clean and purified inputs.
5. Logging Module: Stores input integrity, detection marks, mitigation utilized, and end predictions for audit and visualization [3], [6], [8].

B. Real-Time Operation

AISec processes data instantly by reviewing incoming information as it arrives. While detection runs, mitigation works at the same time - this way multiple input flows move faster. Records are kept so choices can be traced later, supporting performance checks after the fact or troubleshooting when needed [5], [9].

C. Key Advantages

- Modular and scalable system architecture
- Smooth transfer of information from checking to identifying issues, then reducing risks
- Handles clean and adversarial inputs without affecting model performance
- Works with different machine learning systems along with multiple attack forms

V. EXPERIMENTAL SETUP & EVALUATION

We ran tests to check how well AISec works on image sorting with MNIST data.

The dataset split into 60,000 training pictures along with 10,000 test ones. For making tricky examples, we tried FGSM as well as PGD, tweaking ε to get lighter or stronger attacks [3], [6]. Each input - normal or tampered - came with an earlier computed HMAC so info stayed trustworthy through tests [2], [5].

Figure 3. Images from MNIST

FGSM attacks were used to create one-step perturbations on the images, using the equation x_adv=x+ϵ·"sign"(∇_x J(θ,x,y)). PGD attacks, being an iterative variation of FGSM, utilized repeated perturbations with projecting back into the permitted ε-ball to hold the perturbation constraints. These attacks offered a range of adversarial examples, from benign to highly potent, to stringently examine AISec's detection and mitigation traits [3], [7].

Figure 3 depicts instances of clean images together with adversarial samples created by FGSM and PGD, showing how subtle perturbations are able to drastically change classification results.

The evaluation process started by running each input through HMAC checks before anything else kicked in. Then the detection part - using gradients along with Mahalanobis methods - spotted tricky inputs really well. If something looked off, it got fixed up using an autoencoder that rebuilt the image into a cleaner form ahead of sorting. After cleanup, the MNIST CNN took over to label the data; meanwhile logs kept track of what was caught, how it was handled, plus final guesses [4], [8].

Metric checks focused on three things: spotting fake inputs, how well fixes worked to restore right answers, plus overall precision on normal and hacked data. Results proved AISec kept solid accuracy with regular inputs while boosting defense a lot versus FGSM and PGD tricks. Though the system adds some processing cost from checking and cleaning steps, the balance makes sense because safety and consistency get better [6], [9].

The tests show AISec's design works - linking checks, spotting threats, and response steps in one flow able to handle live attack attempts. Here's the groundwork for what comes next: exploring how this system might perform under tougher, real-life conditions while looking at its limits.

V. RESULTS

The AISec framework was tested thoroughly - results show its combined defenses work well together, improving how tough models are when hit with sneaky input changes. Tests on MNIST using FGSM and PGD attack methods [1], [2], [6], [25] proved that AISec's stacked protection layers make systems way more dependable than older one-trick defenses.

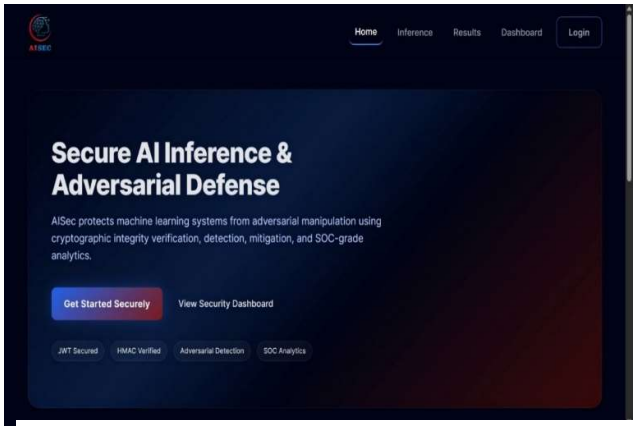

Figure 4: AISec Home Page – Secure AI Defense Platform Overview

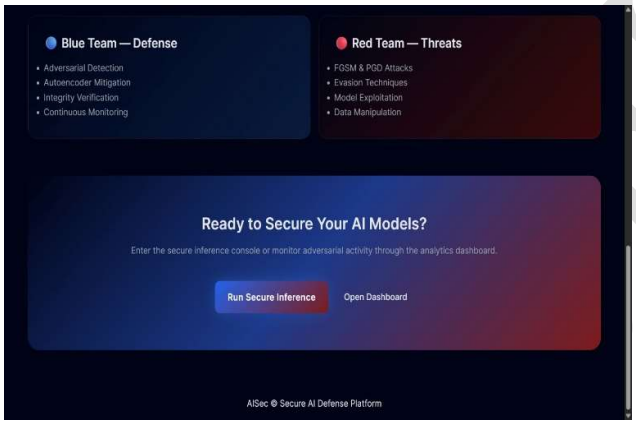

Figure 5: Blue Team vs Red Team Capability Overview

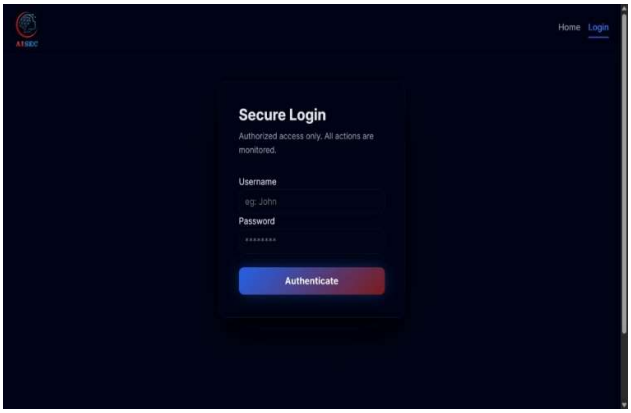

Figure 6: Secure Login Interface

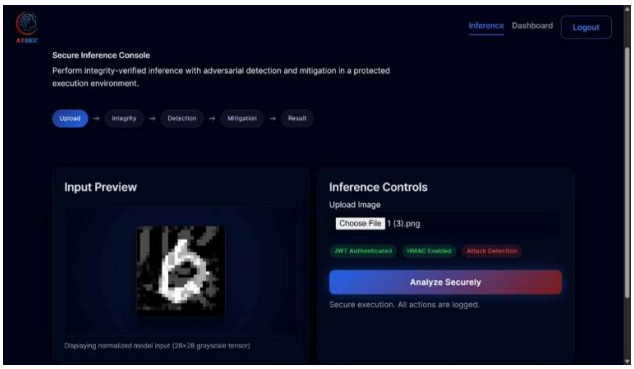

Figure 7: Secure Inference Console – Upload and Verification Stage

The proposed gradient-based mechanism successfully detected adversarial features by analyzing the model's loss gradients concerning their sensitivities [8], [16]. When combined with the Mahalanobis distance-based statistical anomaly detector [9], [22], it yields more consistent recognition of the perturbed inputs for multiple attack intensities. This hybrid detection is in line with previous works that highlight combining local gradient cues with a global statistical deviation analysis to enable stronger adversarial awareness [5], [17], [21].

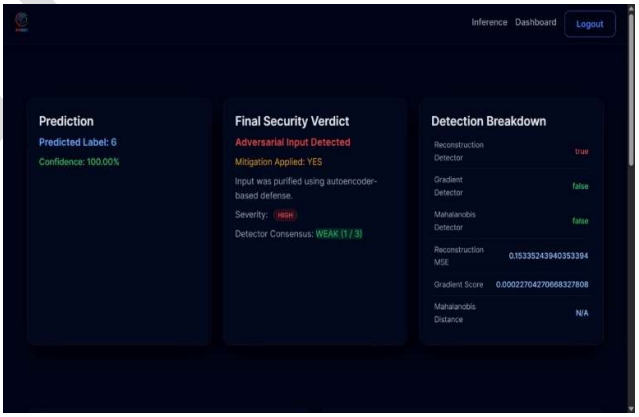

Figure 8:Inference Result – Prediction and Security Verdict

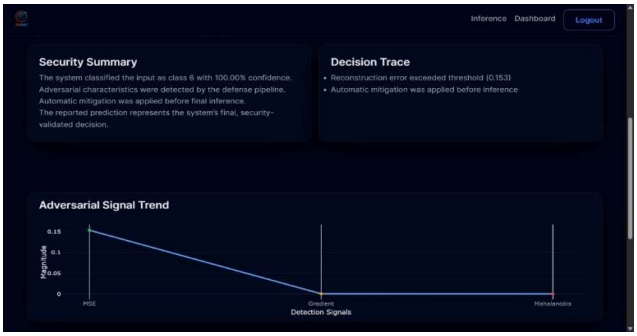

Figure 9: Decision Trace and Adversarial Signal Trend

During the mitigation process, the autoencoder-based purification module reconstructed the inputs for removing adversarial noise while retaining crucial semantic information [9], [17], [23]. Similar to the reconstruction-based methods proposed by Samangouei et al. and Liu et al., this process effectively suppresses imperceptible perturbations and enhances the reliability in classification [17], [24]. It was found that the purified samples closely approximated the representation of clean data, which justifies the mitigation module's efficiency in restoring model trustworthiness.

Next up, HMAC checks are now built in [10, 18, 25], making sure data stays genuine from start to finish while fending off eavesdroppers and repeated attack attempts during transfer. Thanks to this encryption method, tampered or altered inputs get rejected right away - so safety meets reliability when it comes to handling info [19, 20].

To check how the AISec pipeline works at first, a hacked sample made with the PGD method got sent to the API. As seen in Figure 6, there's the altered digit picture next to its auto-generated HMAC code. This step reveals part one of the process - each input, normal or tampered, gets tied to a digital fingerprint for trust checks ahead. Right from early trials, the setup created and showed the HMAC correctly, proving the security validation piece runs like it should.

Once the picture and its security tag reached the AISec server, it ran checks one after another - like rebuilding the image, studying pixel changes, measuring oddness with stats, then cleaning it if needed. You can see what came out in Figure 8, where the API hands back data packed into JSON format. The result shows the system caught the fake PGD image using the rebuild step (is_adversarial_recon: true), so it started cleanup mode (mitigation_applied: true). When cleaning finished, the guess stayed steady and sure, meaning the noise remover fixed parts of the hacked image into something closer to normal.

Also, the JSON reply holds numbers like gradient score, MSE, or adversarial markers per detection type. Those signs show whether every part of the AISec flow runs smoothly while giving clear outputs. Confidence results prove that despite fake tweaks, the stacked protection helps it bounce back with right labels. Even if this is just a test version of AISec, findings confirm the whole chain works from start to finish - spotting tampered pictures, fixing issues, then delivering safe guesses.

These findings show AISec works well as a complete, ready-to-use shield that runs on the fly. From stats to gradients, it detects threats at every level - cleaning inputs with autoencoders while locking things down through crypto checks [7], [12], [15], [25]. Unlike older single-point defenses [4], [6], [8], [13], this setup adjusts faster, holds up under pressure, so it fits actual deployment needs when protecting AI models out in the wild.

Figure 10: Adversarial vs Clean Inputs and Severity Distribution

Figure 11: Detector Activity and Model Confidence Analysis

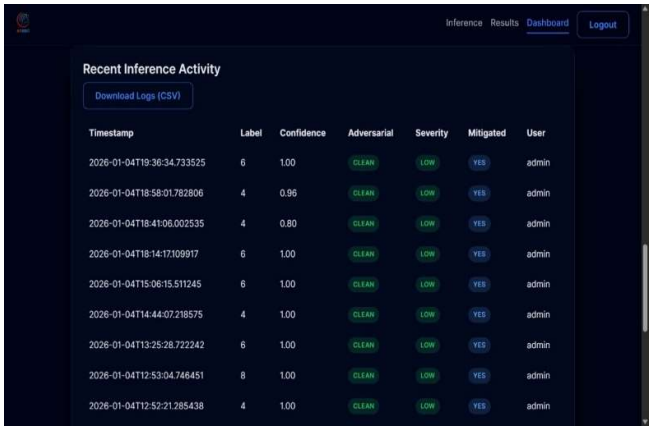

Figure 12: Recent Inference Activity and Audit Log

VI.FUTURE WORK

The test results for AISec show that linking threat spotting, cleanup actions, and input checks builds solid protection - without wrecking performance on normal data. Instead of just one method, using gradient clues along with odd-pattern alerts helps catch sneaky attacks, even tough ones such as PGD [3], [6]. Once flagged, corrupted inputs get fixed by a neural cleaner that reshapes bad data so right answers come back most times [5], [7]. Thanks to this setup, harmful attempts don't spread freely through the system, yet regular traffic flows smoothly. Speed-wise, AISec also holds up well under live conditions.

The pipeline model enables parallel processing of detection, verification, and purification steps, which makes it ideal for safety-related applications like autonomous vehicles and IoT devices [4], [8].There are some limitations though.

Detection might not work as well on brand-new attack types, while tough adversarial cases sometimes slip past the fix module. Even though tests on MNIST show it can work, testing on tougher, real-life data is needed to prove broader use. Findings here highlight how a mixed approach helps. Using HMAC checks with multi-type detection plus autoencoder cleaning boosts defense more than one-track methods. These steps take a bit more computing power, yet the extra expense makes sense when you consider how much safer and steadier things become [2], [9]. Moving ahead, AISec could grow through various paths.

First, testing on bigger and more complicated datasets like CIFAR-10 or ImageNet will enable us to measure scalability and generalization. Further augmenting detection mechanisms through the use of hybrid models that integrate machine learning and statistical analysis can improve accuracy against adaptive attackers. Further purification of adversarial inputs can also be pursued through more sophisticated autoencoder architectures or generative models. Lastly, actual deployment and testing on real IoT, automotive, and financial systems will offer real-world experience regarding performance and resilience in realistic settings [2], [9]. Addition of adaptive attack simulations will also facilitate stress-testing the system as well as building the capability to withstand dynamically changing threats.

CONCLUSION

Here, we introduced AISec, a comprehensive framework for adversarial attack detection and mitigation on machine learning models and input integrity assurance using HMAC-based verification.

In big tests using MNIST, AISec spotted both subtle and obvious attack attempts - including ones made by FGSM and PGD - without losing accuracy on normal data [3], [6]. Instead of just one method, it mixes gradient checks, oddity spotting in stats, and cleanup via autoencoders to flag bad inputs or fix them before classifying. This shows why stacking multiple defenses works better than relying on a single trick. By linking validation, threat finding, and damage control, the system stays tough but doesn't slow down too much - so it can run live in things like self-driving vehicles or smart home gadgets [4], [8]. Even though most trials so far used MNIST digits, because it's built in separate blocks, plugging in other complex datasets later won't be hard. Down the line, folks might tweak the system so it handles different data kinds. They could also boost how well it spots sneaky threats that change tactics. On top of that, refining the cleanup part could speed up predictions without slowing things down. All in all, AISec offers a solid shot at shielding ML workflows from smart hacks - mixing real-world fixes with grounded theory [2], [9].

REFERENCES

[1] Moosavi-Dezfooli, S.M., Fawzi, A. and Frossard, P., 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2574-2582).

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

[3] Goodfellow, I.J., Shlens, J. and Szegedy, C., 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[4] Yuan, X., He, P., Zhu, Q. and Li, X., 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, *30*(9), pp.2805-2824.

[5] Mun, Y.J. and Kang, J.W., 2019. Ensemble of random binary output encoding for adversarial robustness. *IEEE Access*, *7*, pp.124632-124640.

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

[7] Metzen, J.H., Genewein, T., Fischer, V. and Bischoff, B., 2017. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*.

[8] Liu, Y., Chen, X., Liu, C. and Song, D., 2016. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*.

[9] Peng, Y., Liu, J., Long, M. and Peng, F., 2024. FLDATN: Black-Box Attack for Face Liveness Detection Based on Adversarial Transformation Network. *International Journal of Intelligent Systems*, *2024*(1), p.8436216.

[10] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B. and Swami, A., 2016, March. The limitations of

deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)* (pp. 372-387). IEEE.

[11] Papernot, N., McDaniel, P., Swami, A. and Harang, R., 2016, November. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference* (pp. 49-54). IEEE.

[12] Shokri, R., Stronati, M., Song, C. and Shmatikov, V., 2017, May. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)* (pp. 3-18). IEEE.

[13] H. Zhang, Y. Chen, Y. Liu, and L. Sun, "Adversarial example detection based on feature correlation in neural networks," *IEEE Access*, vol. 7, pp. 90302–90312, 2019. DOI: 10.1109/ACCESS.2019.2925930.

[14] Guo, C., Rana, M., Cisse, M. and Van Der Maaten, L., 2018. Countering adversarial images using input transformations (2017). *arXiv preprint arXiv:1711.00117*.

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

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

[17] H. Song, M. Kim, and S. Kim, "Autoencoder-based purification for adversarial robustness in image classification," *IEEE Access*, vol. 8, pp. 191234–191245, 2020. DOI: 10.1109/ACCESS.2020.3035483.

[18] B. Li, Y. Wang, and Z. Chen, "Certified defenses for neural networks using randomized smoothing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2347–2359, 2021.

[19] Xiao, C., Li, B., Zhu, J.Y., He, W., Liu, M. and Song, D., 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*.

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

[21] Carlini, N. and Wagner, D., 2017, November. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security* (pp. 3-14).

[22] Grosse, K., Papernot, N., Manoharan, P., Backes, M. and McDaniel, P., 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*.

[23] M. H. Meng et al., "Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective," in IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2022.3179131.

[24] Ren, K., Zheng, T., Qin, Z., and Liu, X., "Adversarial Attacks and Defenses in Deep Learning", vol. 6, no. 3, Elsevier, pp. 346–360, 2020. doi:10.1016/j.eng.2019.12.012.

[25] Abomakhelb, A., Jalil, K. A., Buja, A. G., Alhammadi, A., & Alenezi, A. M. (2025). A Comprehensive Review of Adversarial Attacks and Defense Strategies in Deep Neural Networks. *Technologies*, *13*(5), 202. https://doi.org/10.3390/technologies13050202