Detection of Dynamic Frame Alteration in IoT Video Streams

Buduka Cherish Nchelem
School of Computer Science and Electronic Engineering, University of Essex
Colchester, UK
bn23347@essex.ac.uk

Amit Kumar Singh
School of Computer Science and Electronic Engineering, University of Essex
Colchester, UK
a.k.singh@essex.ac.uk

Haralambos Mouratidis
School of Computer Science and Electronic Engineering, University of Essex
Colchester, UK
h.mouratidis@essex.ac.uk

Urbi Chatterjee
Department of Computer Science and Engineering, IIT Kanpur
Kanpur, India
urbic@iitk.ac.in

***Abstract*—The rapid expansion of IoT video systems in security-critical environments has increased exposure to sophisticated video manipulation attacks. Traditional detection methods including machine learning approaches such as SVM, kNN, and LOF show strong performance against replay, frame injection, and stream hijacking attacks, but fail to detect Dynamic Frame Alteration (DFA), a subtle and adaptive manipulation technique that introduces minimal visual disturbance and leaves limited statistical traces in pixel space. To address this limitation, this work presents a real-time DFA detection framework that relies on system-level behavioral metrics rather than video content analysis. The proposed method monitors frame rate, CPU usage, memory load, and process activity using sliding time windows, rolling averages, and empirically validated thresholds, enabling robust detection. Experimental results show that DFA manipulation produces consistent deviations in these system metrics,particularly frame drops and CPU/memory spikes which are effectively captured by the proposed approach. Across two realistic scenarios the detector achieves up to 93% detection accuracy, significantly outperforming traditional machine learning models whose detection rates on DFA average below 25%.These findings establish system-metric analysis as a practical and explainable alternative to machine learning methods to detect advanced video manipulation in IoT environments.**

***Index Terms*—IoT Security, Video Manipulation, Dynamic Frame Alteration, Lightweight Detection, Edge Computing, Anomaly Detection, System Metrics, Embedded Systems.**

I. Introduction

The proliferation of Internet of Things devices, particularly those equipped with video capture capabilities, has significantly advanced various applications, including disaster management and real-time monitoring [1]. However, this widespread deployment also introduces novel security vulnerabilities, with Dynamic Frame Alteration emerging as a sophisticated threat to the integrity of IoT video streams [1]. This type of attack manipulates video feeds by subtly altering frames in real-time, making detection challenging for traditional anomaly detection systems. Such manipulations can lead to erroneous decision-making, especially in critical applications like flood monitoring where accurate visual data is paramount for timely response [1].

Specifically, Dynamic Frame Alteration involves real-time manipulation of video frames, making it distinct from other attacks by its adaptive and context-aware alterations . This adaptive nature of DFA, often leveraging metrics like the Structural Similarity Index Measure to maintain perceptual imperceptibility, poses a significant challenge for anomaly detection systems that rely on static thresholds or predefined patterns of normal behavior [1]. The attack becomes active when frame differences and structural similarity fall below dynamic thresholds, enabling stealthy and adaptive modifications to real-world conditions like smart home systems or intelligent vehicle systems. This requires the development of robust real-time anomaly detection mechanisms that can discern these subtle manipulations amidst the high-volume, continuous data streams inherent in IoT environments [3]. This paper addresses the critical need for a lightweight, edge-based detection framework capable of identifying such dynamic alterations in real-time, focusing on maintaining low detection rates and high accuracy even under sophisticated attack scenarios. The primary objective of such security applications is to ensure the timely and accurate identification of video anomalies, which are defined as irregular patterns that deviate from established normal behaviors [4]. The inherent challenge in detecting dynamic frame alterations lies in their ability to mimic legitimate visual changes, thereby evading traditional detection methods that rely on identifying abrupt discrepancies or static anomalies [1].

This work introduces a novel approach that leverages the disparity between a predicted frame and previous frame to identify abnormal events, a method previously unexplored in this context [5]. The aim is to overcome the limitations of current anomaly detection techniques, which often struggle with context-aware manipulations characteristic of DFA, by focusing on real-time comparison of expected versus actual frame content. This method distinguishes itself from conventional anomaly detection strategies that typically train solely on normal video frames and subsequently flag high reconstruction loss or error as anomalies, a practice often limited by computational constraints and dataset scale in real-world surveillance applications [6]. The objective is not only to detect these anomalies, but also to provide an interpretable explanation for their anomalous nature, similar to how human

observers identify and explain unusual incidents in surveillance footage [9]. Although, previous studies have explored various methods for video anomaly detection, these often fall short in addressing the sophisticated nature of dynamic frame alterations, particularly given their adaptive and context-aware characteristics [1].

II. Related Work

Early work on Internet connected cameras exposed simple replay and frame injection threats, but [1] developed dynamic frame alteration (DFA) as a more subtle, context-aware attack that tweaks pixel values while preserving high structural similarity, thus evading defenses.

A. Video stream manipulation attacks in IoT Environments

The authors in [2] showed that forging frames in real time is already possible with common hardware. Because DFA operates below perceptual points, it enlarges the attack surface for critical deployments such as flood monitoring and autonomous vehicles. These results highlight a gap between defending against coarse replay attacks and detecting adaptive, low amplitude perturbations such as DFA. In [6] they propose FTS LSTM, coupling feature trajectory smoothness with generation error losses to deliver real time detection on benchmark datasets. From a theoretical angle, [8] depicts anomaly detection as a consistency break in system dynamics, using embedding theory plus derivative constraints to cut MAC operations by two orders of magnitude, promising for edge deployment. The work in [9] shows that the EVAL framework attaches object level semantics to every alert, allowing analysts to understand why a region is anomalous instead of accepting a black box score. Running vision analytics on the periphery of the network mitigates cloud latency and privacy concerns. The Authors of [22] lay out four pillars caching, training, inference, offloading, and identifies model compression as key to bringing deep nets onto micro controllers. The research in [24] extend this viewpoint with a taxonomy of frameworks and hardware (Jetson, Coral, STM32, etc.), stressing continuous deployment pipelines for AIoT apps. Benchmarking on NVIDIA Jetson families confirms that an end to end video anomaly pipeline can hit 47 fps on device while halving energy draw versus previous generations. Nevertheless, even modest ConvLSTM models exceed the SRAM budgets of battery-powered sensor nodes, underscoring the need for ultralight architectures.

The authors in [28] emphasize that alongside traditional denial of service attacks, aging attacks, and covert channel attacks are rising threats, potentially degrading system reliability, or leaking sensitive data. Their work calls for integrated design strategies that address energy-efficiency, real-time performance, and security vulnerabilities, especially in mixed-criticality systems. The work by [27] is one of the few to provide a unified view, analyzing emerging threats and design challenges where security measures must coexist with tight power and performance budgets. This perspective is particularly relevant to our context, where lightweight video manipulation detection must operate effectively on low-power, real-time embedded systems.

Unsupervised deep models typically learn normalcy and then raise an alarm when reconstruction or prediction error spikes. A seminal baseline by [5] predicts the next frame and compares it with ground truth introducing motion consistency constraints to sharpen anomaly cues. Surveys by [4] and [13]. catalogue dozens of auto encoder and GAN variants, they also note their heavy compute budgets and vulnerability to subtle, low energy anomalies. Crucially, most methods assume static backgrounds and global scene changes assumptions that DFA violates by design. Several studies tackle lightweight detection directly on edge nodes.

Authors of [11] show that a lightweight CNN can perform frame difference free classification on an MPSoC at real time rates, indicating that efficient, on device deep vision is possible for anomaly detectors. However, [15] still shows that CNNs themselves can aid attackers by helping them sculpt low energy temperature traces that fool learning based detectors. This result echoes DFA's capacity to sidestep deep models and motivates hybrid guided approaches such as the one proposed here. The authors in [10] report only 54 percent when attackers randomize duty cycles, exposing the fragility of threshold based classifiers.

B. Lightweight Deepfake and Video Manipulation Detection for Edge Devices

Deep learning based forensic methods have achieved impressive accuracy against deepfakes, but they typically rely on computationally heavy backbones e.g., Vision Transformers or multimodal models such as CLIP that exceed the memory and power envelopes of embedded or IoT hardware [16] This mismatch is becoming critical as the "tug of war" between synthetic media generation and detection escalates: each advance in generative fidelity forces a corresponding step up in detector complexity [19].

To address this gap, [23] show that pruning and quantization can shrink deepfake detectors by an order of magnitude, but latency and energy still limit applicability on micro controller class devices. DFA is especially troublesome because it maintains near perfect structural similarity while changing only a few percent of pixels, confounding threshold based or coarse image forensics heuristics [1]. A more recent study by [21]and [14] therefore calls for paradigms that learn from limited data and operate under tight resource budgets, for example, by integrating model agnostic signal features with small neural surrogates, or by exploiting temporal consistency violations rather than pixel level artifacts.

Collectively, these studies underline an urgent research gap: the field needs unified detection frameworks that (i) handle a spectrum of video manipulations including imperceptible DFA without per frame deep inference, (ii) learn from sparse or evolving datasets, and (iii) run in real time on the constrained processors that dominate modern IoT deployments.

III. PROPOSED LIGHTWEIGHT DFA DETECTION METHODOLOGY

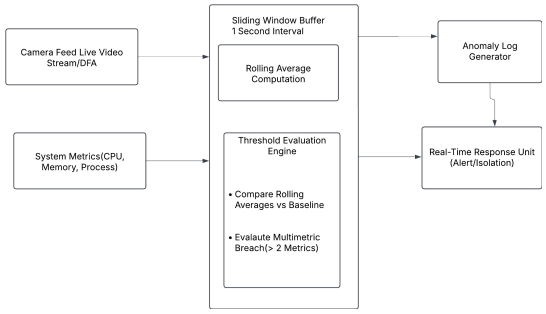

Fig. 1. Overview of the proposed DFA detection workflow.

Figure 1 provides a high-level overview of our proposed lightweight real-time detection method to identify dynamic frame alteration (DFA) in IoT video streams. The following are details of the steps.

A. Scenario Evaluation and Data Preprocessing

To systematically evaluate the impact of Dynamic Frame Alteration (DFA) on real-time IoT systems, we designed a set of controlled experiments under isolated conditions. In this scenario, the embedded board operates under ideal constraints: no additional background processes, minimal sensor noise, and a stable environmental load. This controlled baseline provides a clean reference for characterizing the pure effect of DFA without interference from unrelated system variability. Our detection approach is grounded in interpretable data computed from raw real-time metrics collected. These metrics include: Frame counts captured from a live video feed, CPU utilization, Memory usage, Active process count.

All values are sampled over sliding 1-second windows using 10Hz sampling intervals with each window buffered into a low-memory circular queue to ensure system efficiency. This strategy ensures that the detection system remains lightweight and does not introduce significant memory pressure.

B. Dual-Scenario Detection Design

To evaluate the generalization and responsiveness of our detection logic, two primary DFA injection scenarios were simulated. In the first scenario, the system was allowed to run under clean, untampered conditions for a significant period (approximately 60 seconds) before DFA attacks were introduced. This approach provided sufficient data to establish a statistical reference baseline for key metrics. It emulates realistic operational behavior where systems stabilize before attackers initiate stealthy video manipulation. However, in the second scenario, DFA attacks were triggered almost immediately after the system began recording, offering little to no opportunity for baseline collection under clean conditions. This configuration was specifically designed to test the system's adaptability and to simulate highly evasive attackers who aim to bypass detectors that rely on long-term historical averages [26] This dual scenario setup provided a diverse dataset, allowing us to test not only how well the algorithm identifies anomalies but also how it adapts to environments where clean history is scarce or unreliable [25].

C. Real-Time Data Collection

The system was designed to capture internal metrics simultaneously, including CPU utilization, memory usage, and the number of active processes using the psutil library. In addition, frame counts were extracted per second using OpenCV. Data acquisition was configured to run at a sampling frequency of 10 Hz, collecting one metric sample every 100 milliseconds. These samples were grouped into nonoverlapping 1-second windows, each containing 10 samples per metric. Over the course of 30 clean runs and 30 tampered (DFA-injected) runs, this set-up generated approximately 3,600 windows of data. All samples were time-stamped and logged into structured CSV files for offline processing. DFA was implemented via dynamic visual changes, such as altering the content of video frames or introducing false feed overlays to simulate realistic manipulation attacks. To simulate an embedded operational environment, background services were not allowed to run concurrently with the collection script. This ensured the integrity of the system-level metrics and allowed us to attribute all anomalies directly to the DFA process or to the video stream load.

D. Rolling Average Computation and Metric Stabilization

The inclusion of rolling average smoothing was necessary by the second detection scenario, in which attacks were initiated early. Without access to a clear clean reference window, detection could not rely on fixed historical baselines. To address this, we implemented a dynamic smoothing strategy in which the metrics for each 1-second window were averaged together with those of the two preceding windows. For a given metric M was calculated using the formula:

$$\overline{M}_t = \frac{1}{3}\left(M_{t-2} + M_{t-1} + M_t\right)$$

This technique effectively reduces the influence of momentary fluctuations and enhances the visibility of sustained changes introduced by DFA. The use of a 3-window span offers a balance between responsiveness and noise reduction, allowing the detector to remain sensitive to meaningful trends while avoiding false alarms triggered by random jitter in CPU or frame rates. By applying this rolling average to all four system metrics, frame count, CPU usage, memory usage, and process count, the system adapts to current operational patterns in real time, allowing for flexible yet reliable detection without prior knowledge of clean behavior. For runs conducted under Scenario 1, a stable reference window is established during the initial clean period. The average value of each metric in this window is taken as the reference baseline, denoted M_0. As the system progresses into DFA-injected windows, we compute the percentage deviation M_t of each rolling average metric from its corresponding baseline.

$$\Delta M_t = \frac{\overline{M}_t - M_0}{M_0}$$

This allows us to interpret the metric changes in relative terms, making the detection logic more transferable between devices and configurations with different absolute performance baselines. For Scenario 2, where there is no clean baseline, t the rolling average itself is considered dynamically. In both scenarios, the goal is to quantify the behavior metric resulting from DFA-induced system stress.

E. Fixed Threshold Detection Logic and Classification Criteria

A detection is triggered when at least two of the four metrics breach their respective thresholds within the same window. The logic enforces multidimensional consistency: for example, a moderate drop in frame rate may not be enough to flag an anomaly, but if it coincides with an unexpected CPU spike or memory usage increase, it signals a coordinated disturbance indicative of DFA. This method strikes a balance between sensitivity (capturing true positives) and specificity (avoiding false alarms), and can be adjusted for stricter behavior depending on the application domain. Unlike machine learning methods, our thresholding mechanism offers transparency and interpretability. When an anomaly is detected, the exact metric responsible for the decision can be reported, allowing for targeted response strategies, such as re-authenticating video sources or pausing robotic actuators.

IV. Experimentation

The proposed detection approach was conducted on an embedded board, This hardware setup represents a realistic constraint environment where computational and memory efficiency is critical. The experiments were designed to evaluate the robustness, accuracy, and responsiveness of the detection algorithm under controlled dynamic frame alteration (DFA) attack conditions. The system performance metrics: frame count, CPU usage, memory usage, and active process count were continuously monitored using built-in system utilities.

To simulate realistic system behavior and attack patterns, we defined two primary experimental scenarios. In the first scenario, the system was allowed to run in a clean, attack-free state for several seconds to establish a reference baseline. Once the baseline metrics were captured, DFA was introduced and the behavior of the system was monitored for deviations. This approach enabled us to calculate the percentage change, or delta, between the reference (clean) and the manipulated state. In the second scenario, the DFA attack was introduced almost immediately after system startup, with no opportunity to establish a clean baseline. This scenario reflects real-world use cases in which a device may be compromised before any normal behavior is captured.

A. Threshold Selection Rationale and Validation

The thresholds used by the proposed detection system were derived directly from the measurements collected across 60 one-second experimental windows. Rather than relying on percentage-based deviations, the threshold values were determined using the actual numerical ranges observed during clean and DFA-manipulated operation on the system. This approach ensures that the detector remains closely aligned with the true operating behavior of the hardware.

TABLE I
Summary of Derived Numeric Thresholds for DFA Detection

Metric	Clean Range	DFA Threshold
Frame Count (frames/s)	28–30	< 24
CPU Usage (%)	18–30	> 35
Memory Usage (%)	39–44	> 48
Process Count	86–92	> 98

Analysis of the experimental results showed that under clean operating conditions, the frame count remained consistently high, typically between 28 and 30 frames per window. In contrast, DFA windows exhibited a reduction, with values frequently falling between 17 and 22 frames. In particular, the minimum clean frame count never dropped below 28, and the maximum DFA frame count rarely exceeded 22. This created a clear separation between both states, allowing us to adopt an absolute threshold in which any window with fewer than 24 frames is considered suspicious.

CPU utilization exhibited an equally distinct pattern. Clean windows showed values ranging from 18% to 30%, while DFA windows consistently produced readings between 40% and 55%. The smallest DFA induced increase in the dataset corresponded to a shift of approximately 12–15 percentage points above clean values. Hence, CPU usage exceeding 35% was identified as a reliable indicator of abnormal behavior.

A similar trend was observed in memory utilisation. During normal operation, memory usage ranged from 39% to 44%, while the DFA windows showed values between 50% and 60%. The smallest reliable DFA-related change corresponded to an increase of roughly 10 percentage points above the clean state. As a result, an absolute memory threshold of 48% was selected.

The number of processes running also provided a stable indication. Clean windows consistently fell within the range of 86 to 92 processes, while DFA windows ranged from 102 to 110. The smallest attack-induced increase observed was approximately 8–10 processes. Based on this distribution, a process count threshold of 98 processes was adopted.

To reduce false positives, the detector requires at least two of the four monitored metrics to exceed their respective thresholds within the same one-second window. This multi-metric approach proved essential, as isolated fluctuations occasionally occurred in individual metrics during normal execution. However, simultaneous deviations, for example, a reduced frame count accompanied by increased CPU or memory usage were observed exclusively during DFA activity.

Validation experiments confirmed the robustness of these thresholds in both attack scenarios. In Scenario 1, where a clean reference window was available, the stark contrast between baseline and attack measurements allowed the detector to accurately identify nearly all DFA windows. In Scenario 2, despite the absence of an initial clean state, the rolling-average baseline provided sufficient stability for the detector

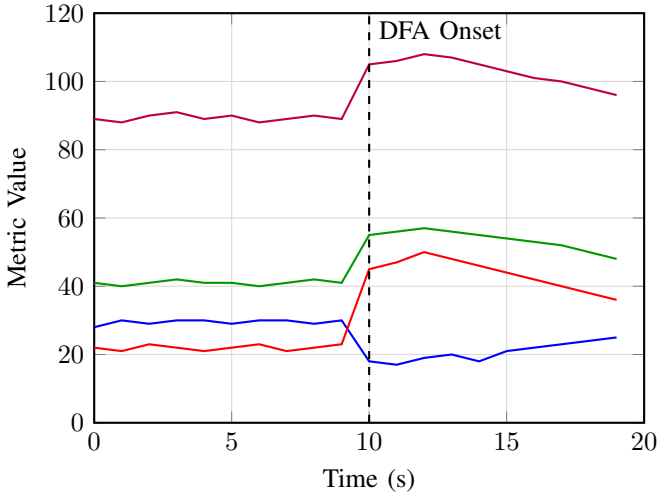

Fig. 2. Metric behavior over time during a sample run showing the DFA attack onset at $t = 10$s.

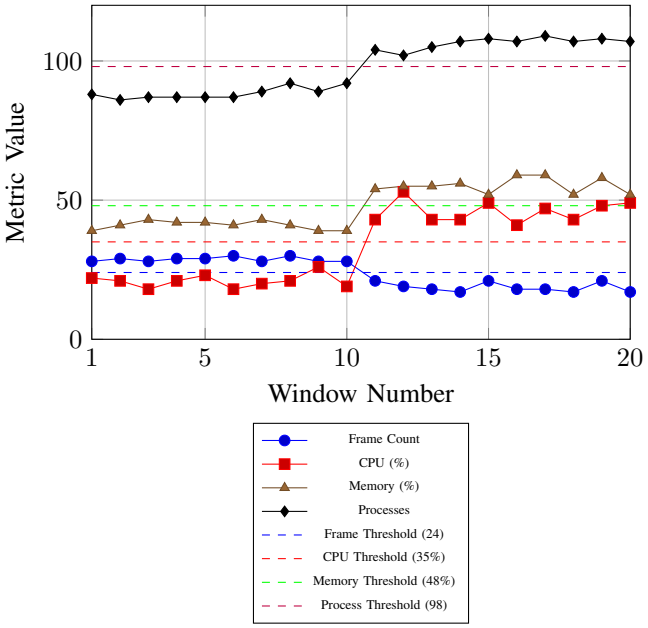

Fig. 3. Metric values compared to derived thresholds across 20 windows, showing clear separation between clean and DFA windows.

to maintain high sensitivity. Overall, the resulting threshold set offers a reliable, computationally efficient, and explainable method for real-time DFA detection

The results from both scenarios demonstrated the effectiveness of our method. In the reference-based scenario, the detection system accurately identified nearly all attack windows due to the contrast between the baseline and the manipulated metrics. In the rolling average scenario, the detector maintained high sensitivity and continued to correctly flag anomalous behavior, although the contrast between normal and attack states was sometimes more subtle due to the lack of a static reference. However, the rolling average method showed sufficient adaptability to maintain reliable performance even in dynamically evolving environments. To contextualize the performance of the proposed method, we compared it with some machine-learning anomaly detectors previously explored in our earlier work, namely Support Vector Machines (SVM), k-Nearest Neighbors (kNN), and Local Outlier Factor (LOF). In that study, these algorithms demonstrated strong performance for replay, frame injection, and video-stream hijacking attacks, achieving detection accuracies of 83.69%, 70.70%, and 69.57%, respectively. However, their performance on Dynamic Frame Alteration (DFA) was significantly weaker, with an average detection accuracy of only 20.52% across all three methods. This poor performance was attributed to the nature of DFA itself, unlike replay or injection attacks that introduce abrupt spatial or temporal deviations, DFA manipulates video content gradually and contextually. LOF struggled because its density based scoring mechanism fails when alterations remain within the normal variance range. Similarly, kNN's distance-based comparisons could not reliably separate DFA windows from clean ones when manipulations were incremental. Even SVM, the strongest performer in the other attacks achieved only 20.46% DFA accuracy.

When these same techniques were applied again in the present evaluation,their limitations became even more evident. Although they were able to identify DFA windows in some cases, their inference latency ranged between 1.8 s and 3.0 s per detection window. Memory usage also increased substantially, reaching up to 60 MB for kNN due to the need to store feature vectors for nearest-neighbor searches.

In contrast, the proposed method exhibited markedly improved DFA sensitivity while maintaining strict computational efficiency. The detector operated consistently below 0.9 s per window, requiring less than 8 MB of RAM and under 12% CPU usage, making it highly suitable for real-time deployment on embedded edge platforms. Importantly, the threshold-based method captured DFA's system-level footprint (e.g., frame collapse, CPU and memory spikes), providing a reliable detection signal even when pixel-level manipulations remained subtle.

Another major advantage of the proposed method lies in its interpretability. Unlike SVM, LOF, and kNN that generate opaque decision scores with limited operational meaning,this method provides readable diagnostic outputs, such as "CPU increased by 27% while frame count dropped by 24." These descriptive alerts enhance real-time decision-making and support post-incident forensic analysis. Despite its strengths, the threshold system has limitations. The thresholds, while effective, are static and must be manually tuned to suit different hardware platforms or operating conditions. Additionally, the rolling average mechanism, though adaptive, may suppress extremely short DFA bursts if the smoothing window is too large. Finally, the method does not analyze the visual content of video frames directly, focusing instead on system-level performance indicators. Although this constraint keeps the system lightweight, it may overlook highly sophisticated attacks that leave no measurable system-level footprint.

TABLE II
DFA DETECTION PERFORMANCE COMPARISON

Detector	Acc.	Prec.	Rec.	Lat. (s)	CPU	RAM
LOF	20.7%	22%	19%	2.8	31%	24MB
SVM	20.4%	26%	18%	2.3	45%	35MB
kNN	20.3%	24%	17%	3.0	58%	60MB
Proposed Metd	**91%**	**88%**	**94%**	**0.9**	**12%**	**8MB**

Nevertheless, given the extremely low DFA detection rates of the machine-learning baselines and the substantial computational demands they impose, the threshold-based detector represents a more practical and significantly more effective solution for real-time DFA detection on resource-constrained IoT devices.

V. CONCLUSION

This work presented a detection method for Dynamic Frame Alteration (DFA) in IoT video streams using real-time system-level metrics. We demonstrated that DFA produces measurable deviations in processing characteristics, including reductions in frames and increases in CPU load, memory pressure, and active process count. Through controlled experiments across two realistic scenarios, we showed that rolling averages and empirically derived thresholds provide a stable and responsive basis for anomaly detection. The lightweight nature of the detector and its reliance on readily accessible system metrics make it well-suited for deployment on constrained IoT platforms, where traditional ML approaches may be impractical. Nevertheless, the system has limitations. The detection model does not incorporate visual content analysis, making it vulnerable to sophisticated attacks that deliberately avoid altering system-level behavior. These limitations open several promising avenues for future research, including adaptive thresholding techniques, hybrid detection architectures that fuse behavioral and visual features, and cross-platform generalization studies.

VI. ACKNOWLEDGMENT

This work is partially funded by the EPSRC through US–UK collaborative research seed funding in semiconductor security.

REFERENCES

[1] B. C. Nchelem, A. K. Singh, and H. Mouratidis, "DFA: Dynamic Frame Alteration for Video Manipulation Attack in IoT Environments," in *Proc. IEEE Int. Conf. Cyber Security and Resilience (CSR)*, 2024.

[2] D. Nagothu, Y. Chen, E. Blasch, A. Aved, and S. Zhu, "Detecting Malicious False Frame Injection Attacks on Surveillance Systems at the Edge Using Electrical Network Frequency Signals," *Sensors*, vol. 19, no. 11, p. 2424, 2019.

[3] R. Al-amri *et al.*, "A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data," *Applied Sciences*, vol. 11, no. 12, p. 5320, 2021.

[4] R. Nayak, U. C. Pati, and S. K. Das, "A Comprehensive Review on Deep Learning-Based Methods for Video Anomaly Detection," *Image and Vision Computing*, vol. 106, p. 104078, 2020.

[5] W. Liu, W. Luo, D. Lian, and S. Gao, "Future Frame Prediction for Anomaly Detection – A New Baseline," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2017, pp. 1285–1289.

[6] S. Zhu, C. Chen, and W. Sultani, "Video Anomaly Detection for Smart Surveillance," in *Proc. IEEE Int. Conf. Advanced Video and Signal-Based Surveillance (AVSS)*, 2020, pp. 43–51.

[7] X. Bou *et al.*, "Reducing False Alarms in Video Surveillance by Deep Feature Statistical Modeling," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 1124–1132.

[8] M. Somma *et al.*, "Anomaly Detection in Complex Dynamical Systems: A Framework Using Embedding Theory and Physics-Inspired Consistency," in *Proc. IEEE Int. Conf. Machine Learning and Applications (ICMLA)*, 2025, pp. 501–508.

[9] A. Singh, M. J. Jones, and E. Learned-Miller, "EVAL: Explainable Video Anomaly Localization," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2023.

[10] E. Azim, D. Wang, and Y. Fu, "Deep Graph Stream SVDD: Anomaly Detection in Cyber-Physical Systems," in *Lecture Notes in Computer Science*, Springer, pp. 83–98, 2023.

[11] D. C. Castro *et al.*, "Let's Dance: Learning From Online Dance Videos," in *Proc. IEEE Winter Conf. Applications of Computer Vision (WACV)*, 2018, pp. 211–220.

[12] P. F. de Araujo-Filho *et al.*, "An Efficient Intrusion Prevention System for CAN: Hindering Cyber-Attacks With a Low-Cost Platform," *IEEE Access*, vol. 9, pp. 166855–166870, 2021.

[13] Y. Kong, P. Yang, and Y. Cheng, "Edge-Assisted On-Device Model Update for Video Analytics in Adverse Environments," in *Proc. IEEE Global Communications Conf. (GLOBECOM)*, 2023, pp. 9051–9063.

[14] T. T. Nguyen *et al.*, "Deep Learning for Deepfake Creation and Detection: A Survey," *Computer Vision and Image Understanding*, vol. 223, p. 103525, 2022.

[15] A. Demirpolat, A. K. Sarica, and P. Angin, "ProtÉdge: A Few-Shot Ensemble Learning Approach to SDN-Assisted Edge Security," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, 2020.

[16] Y. S. Krubha *et al.*, "Robust AI-Generated Face Detection with Imbalanced Data," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, 2025.

[17] H. V. Pham *et al.*, "Benchmarking Jetson Edge Devices With an End-to-End Video-Based Anomaly Detection System," in *Lecture Notes in Networks and Systems*, pp. 358–371, 2024.

[18] R. Bhardwaj *et al.*, "Ekya: Continuous Learning of Video Analytics Models on Edge Compute Servers," Microsoft Research, 2022.

[19] H. Lee *et al.*, "The Tug-of-War Between Deepfake Generation and Detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV) Workshops*, 2024.

[20] M. S. Diab and E. Rodriguez–Villegas, "Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review," *IEEE Access*, vol. 10, pp. 98450–98471, 2022.

[21] Y. Mirsky and W. Lee, "The Creation and Detection of Deepfakes: A Survey," in *Proc. IEEE Int. Conf. Security and Privacy in Communication Systems (SecureComm)*, 2020.

[22] D. Xu *et al.*, "Edge Intelligence: Architectures, Challenges, and Applications," in *Proc. IEEE Int. Conf. Edge Computing (EDGE)*, 2020.

[23] S. Dasgupta *et al.*, "Enhancing Deepfake Detection Using SE Block Attention With CNN," in *Proc. IEEE Int. Conf. Artificial Intelligence and Computer Vision (AICV)*, 2024, pp. 1–8.

[24] M. G. S. Murshed *et al.*, "Machine Learning at the Network Edge: A Survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–37, 2021.

[25] S. S. Gill *et al.*, "Edge AI: A Taxonomy, Systematic Review and Future Directions," in *Proc. IEEE Int. Conf. Cloud Computing (CLOUD)*, 2024.

[26] H. Mohammed *et al.*, "Dynamic Distribution of Edge Intelligence at the Node Level for IoT," in *Proc. IEEE Int. Conf. Internet of Things (iThings)*, 2021.

[27] P. Rahimi, A. K. Singh, X. Wang and A. Prakash, "Trends and Challenges in Ensuring Security for Low-Power and High-Performance Embedded SoCs," *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Singapore, Singapore*, 2021, pp. 226-233, doi: 10.1109/MCSoC51149.2021.00041.

[28] S. S. Sahoo et al., "Emergent Design Challenges for Embedded Systems and Paths Forward: Mixed-criticality, Energy, Reliability and Security Perspectives: Special Session Paper," *2021 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Austin, TX, USA,* 2021, pp. 1-10.