iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81859

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

A Multi-Modal Deep Learning Framework for Pixel-Level Fault Detection and Hardware Component Isolation in Satellite Imagery

Pasupuleti Koushik Kumar[1], Pasupuleti Naga Lakshmi Priyanka[2], Beemanapalli Rama Chandra[3], Chenna Satyannarayana[4], Banavath Venkata Ram[5]
Department of Computer Science and Engineering (AIML), Acharya Nagarjuna University, Guntur, India

Abstract: ***Satellites are increasingly mission-critical infrastructure, yet the tools used to monitor their health have not kept pace with the complexity of modern imaging sensors. Most operational monitoring systems still depend on telemetry thresholds a method that works well for obvious failures but consistently misses the slow, subtle degradation that begins at the pixel level. This paper describes SmartSAT Guardian, a system we built to address exactly that gap. Rather than waiting for a telemetry channel to cross a threshold, SmartSAT Guardian analyses the images a satellite captures and looks for pixel-level signatures that indicate specific hardware problems. When it finds one, it identifies not just the fault type but the physical component responsible — the photodetector array, the readout circuit, the lens mechanism, or the data downlink module. The system combines a CNN Autoencoder for anomaly detection, an ANN for fault classification, an LSTM for tracking how faults evolve over time, Granger Causality for confirming root cause through telemetry, and an NLP module that turns all of this into a readable report for ground operators. On our synthetic test dataset with five fault categories, the ANN classifier reached 94.1% accuracy with an F1-score of 92.5%. The entire pipeline runs as a decentralised web application on the Internet Computer Protocol, with no dependency on traditional cloud hosting.***

Index Terms: ***satellite fault detection, pixel anomaly detection, CNN autoencoder, LSTM temporal tracking, Granger causality, hardware component isolation, Internet Computer Protocol.***

I. INTRODUCTION

Satellites are not easy things to fix. Once one is in orbit, there is no sending a technician to replace a burned-out component or recalibrate a drifting sensor. Ground operators have to work with whatever data the satellite sends back — and if that data is telling them something is wrong, they need to fig-ure out what and how serious before it becomes catastrophic

The standard approach to satellite health monitoring has been around for decades: collect telemetry — temperature read-ings, battery voltage, signal strength, altitude — and set threshold alerts for when something drifts out of the expected range. This works fine for catching severe, sudden failures. What it does not catch is the slow kind of degradation that shows up first in image quality, not in any single telemetry channel.

A photodetector that is starting to fail will produce dead pixels [9], [10]. Long before any temperature or voltage reading looks unusual.

Data transmission error will scatter noise across an image frame [9], [10]. well before the signal strength drops enough to trigger an alert [1].That time gap — between when a fault first appears in the image and when telemetry catches it — is exactly the window SmartSAT Guardian is designed to exploit. By treating the satellite's own images as a diagnostic signal, the system can detect problems earlier, identify which hardware component is responsible, and give operators a concrete recommendation instead of a vague warning.

The system does this through five interconnected components working in sequence: a CNN Autoencoder spots abnormal pixel patterns, an ANN classifier identifies the fault type and maps it to the failing hardware, an LSTM tracks how the fault progresses across image frames, Granger Causality confirms the propagation path through telemetry channels, and an NLP module writes up a plain-language report. Everything runs on the Internet Com- puter Protocol, which means no central server to go down and no cloud vendor to depend on.

II. LITERATURE REVIEW

Work on satellite fault detection and image-based anomaly detection has been scattered across several fields. Looking across the existing literature, four broad research threads stand out — and all of them fall short of what SmartSAT Guardian attempts.

A. *Telemetry-Based Anomaly Detection*

The most mature line of work uses LSTM networks to find anomalies in spacecraft telemetry. Hundman et al. [2] built a well-known system using NASA SMAP and MSL telemetry data, with dynamic thresholding that performed well on time series anomalies. Malhotra et al. [3] showed that LSTM encoder-decoders could use reconstruction error as an unsupervised fault signal across multiple sensor channels. Both are solid approaches but they work on scalar telemetry values and never look at image data at all.

B. *CNN-Based Image Analysis*

CNN autoencoders have been applied to satellite imagery, though not for hardware fault detection. Chen et al. [4] used them for hyperspectral feature extraction, which validated the basic architecture for satellite image processing. Zhu et al. [5] surveyed deep learning across remote sensing tasks more broadly, covering classification and object detection. Neither study was asking the question we are asking: what does a pixel anomaly tell you about the hardware that produced it?

C. *Satellite FDIR Systems*

Fuertes et al. [6] applied SVMs and Random Forest classifiers to satellite Fault Detection, Isolation, and Recovery (FDIR). Their system got reasonably far with telemetry features alone — partial component isolation was possible — but image data was never part of the input. The fault signal was still telemetry.

D. *Causal Inference in Fault Analysis*

Bauer et al. [7] used Granger Causality and transfer entropy to map how disturbances propagate through industrial control systems. That methodology translated well into the causal confirmation module of SmartSAT Guardian, adapted to show how a fault in one satellite subsystem triggers cascading effects in others

E. *Limitations of Existing Systems*

After reviewing this body of work, the gap is clear: there is no existing system that combines pixel-level image fault detection, hardware component attribution, temporal degradation tracking, causal inference, natural language reporting, and decentralised deployment in one unified pipeline. Table I shows how SmartSAT Guardian compares against the closest prior systems.

Table I: Comparison With Related Systems

System	Method	Image-Based	Component Isolation	NLP Report	Decentralized
Hundman et al. [2], 2018	LSTM Auto-encoder	No	No	No	No
Fuertes et al. [6], 2016	SVM / Random Forest	No	Partial	No	No
Chen et al. [4], 2016	CNN Auto-encoder	Yes	No	No	No
Zhu et al. [5], 2017	CNN Classifier	Yes	No	No	No
SmartSAT Guardian (Proposed)	CNN AE + ANN + LSTM + Granger + NLP	Yes	Yes	Yes	

III. SYSTEM ARCHITECTURE

A. *Design Principles*

Four ideas shaped every design decision in this system.

The first is that images should come first. Telemetry is useful, but imaging sensors degrade in ways that show up in pixels before they show up in any scalar measurement. The system, therefore, treats satellite imagery as the primary fault signal and uses telemetry only to confirm what the image analysis already found.

The second is hardware attribution. Knowing that something is wrong is not enough. An operator needs to know which physical component to look at — the photodetector array, the readout circuit, or the lens. Every fault the system detects gets traced to a specific component.

The third is temporal awareness. A single anomalous frame could be noise. A trend across ten frames is evidence of real degradation. The system tracks how fault characteristics evolve to estimate both how serious the problem is and how long before it becomes critical.

The fourth is decentralised deployment. Running on the Internet Computer Protocol means the monitoring system itself cannot go down due to a cloud outage, cannot be tampered with, and is always reachable regardless of what happens to any particular data centre

B. *Layered Architecture*

The system processes each satellite image through six sequential stages:

1) Input: Satellite images (128×128 grayscale) and raw telemetry streams are ingested simultaneously. Images are normalised to [0, 1] and telemetry channels are standardised before any processing begins, ensuring consistent input distributions across all pipeline stages.

2) Classification: The CNN Autoencoder processes the normalised image and produces a pixel-wise anomaly heatmap. A reconstruction error map is computed and thresholded at the 95th percentile to generate a binary fault mask.

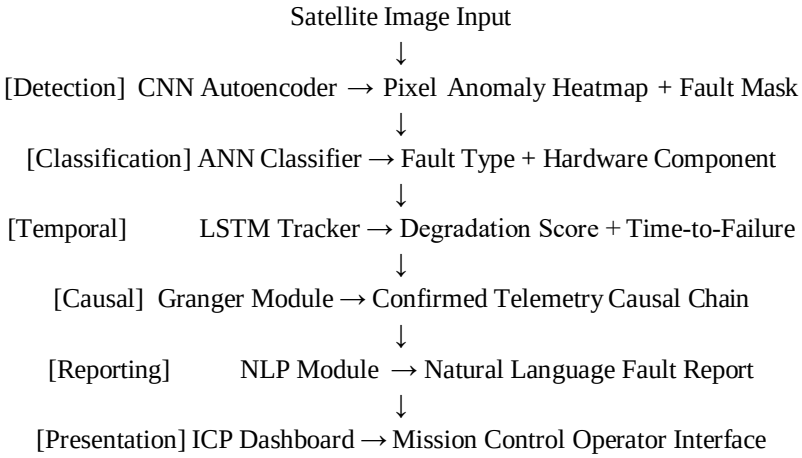
3) Temporal Tracking: The ANN Classifier receives spatial features extracted from the fault mask — faulty pixel count, centroid coordinates, regional variance, and orientation score — and identifies the fault type.

4) Causal Confirmation: VAR model and Granger Causality tests trace the fault propagation path through telemetry

5) Reporting: NLP module generates a plain-language fault report

6) Presentation: Mission control dashboard on ICP delivers the output to operators

Fig. 1:SmartSAT Guardian End-to-End System Architecture



C. *Hardware Component Mapping*

Each fault type maps to a specific hardware component based on the physical mechanism that produces it. Table II documents this mapping.

Table II: Fault-to-Hardware Component Mapping

Fault Class	**Hardware Component**	**Physical Mechanism**
Dead Pixels	Photodetector Array	Failed photosensitive elements
Hot Pixels	Thermal Regulation Unit	Thermal runaway in the sensor
Fault Class	Hardware Component	Physical Mechanism
Stripe Artifact	Sensor Array Readout Circuit	Bus failure in the readout circuit
Blur Patch	Lens / Focus Mechanism	Mechanical misalignment or contamination
Salt & Pepper	Data Transmission Module	Bit errors during image downlink

D. Technology Stack

The AI/ML side runs in Python 3.9+ using TensorFlow/Keras 2.x [12] for the CNN, ANN, and LSTM models; NumPy for image array operations and fault injection; Pandas for telemetry handling; scikit-learn for normalisation and metrics; statsmodels for the VAR model and Granger tests; and Matplotlib for heatmap visualisation. The frontend is built with React 18, TypeScript, and Tailwind CSS, using Recharts for all chart components. The backend is written in Motoko and deployed as a canister on the Internet Computer Protocol [8].

IV. METHODOLOGY

The SmartSAT Guardian pipeline applies five complementary techniques in sequence to detect, classify, track, and report hardware faults in satellite imagery. Each technique is selected for its theoretical suitability to a specific stage of the fault detection problem.

A. CNN Autoencoder — Pixel Anomaly Detection

A Convolutional Neural Network (CNN) Autoencoder [4], [14] is...an unsupervised deep learning model that learns to compress an input image into a low-dimensional latent representation and then reconstruct it. When trained exclusively on fault-free satellite images, the model learns the statistical distribution of normal pixel patterns. Autoencoder-based anomaly detection methods have been widely used due to their ability to learn normal data distributions and detect deviations [14]. This reconstruction error works as a spatially precise anomaly signal — and crucially, it requires no labelled fault examples during training. The system only needs to know what a healthy image looks like. The resulting error map is then thresholded to produce a binary fault mask that marks the exact pixel locations of detected anomalies.

B. ANN Fault Classifier — Hardware Attribution

An Artificial Neural Network (ANN) is a supervised classification model [13]. that learns a mapping from input features to output labels through a series of weighted, nonlinear transformations. Once the CNN Autoencoder has identified where the fault is, the ANN takes over to identify what kind of fault it is. Spatial features describing the shape, size, and distribution of the detected anomaly region are extracted from the fault mask and fed into the ANN. The network outputs a classification across five fault categories — dead pixels, hot pixels, stripe artefact, blur patch, or salt-and-pepper noise — along with a confidence score from the Softmax output layer. The predicted fault class is then passed through the deterministic rule table from Table II to identify the responsible hardware component.

C. LSTM Temporal Degradation Tracker

A single image frame tells you that a fault exists right now. What it cannot tell you is whether the fault appeared yesterday and has barely changed, or whether it has been growing rapidly for the past hour. That distinction matters enormously for how urgently an operator needs to respond. A Long Short-Term Memory (LSTM) network [2] processes addresses this by processing fault-severity features extracted from 10 consecutive image frames as a time sequence. Because LSTM units maintain internal state across time steps, the network can learn what a stable fault looks like versus one that is accelerating. The output is a degradation score between 0 and 1, from which a time-to-failure estimate is derived. This kind of temporal pattern recognition is simply not possible with single-frame methods.

D. Granger Causality — Root Cause Analysis

Granger Causality [7] is a statistical method for determining whether one time series genuinely helps predict another, beyond what the second series already predicts from its own history. It is not about correlation — two channels can be correlated without one causing the other. Granger Causality tests for directed, time-lagged influence

In this system, a Vector Autoregression (VAR) model is fit to four telemetry channels — temperature, battery voltage, signal strength, and orbital altitude. The Granger tests then identify which channels are driving changes in others, producing a directed causal graph. This confirms the propagation path of the fault across hardware subsystems and distinguishes the original fault source from downstream effects.

E. NLP Fault Report Generation

The final stage converts everything the pipeline has produced — fault type, hardware component, pixel count, confidence score, degradation score, time-to-failure — into a plain-language report that a ground operator can read and act on without needing to interpret numbers or graphs.

Table III: Severity Classification and Recommended Actions

Severity	Condition	Recommended Action
CRITICAL	N_f > 1000 or d > 0.8	Immediate inspection
HIGH	N_f > 500 or d > 0.6	Inspection within 24 Hours
MEDIUM	N_f > 200 or d > 0.4	Schedule maintenance
LOW	Otherwise	Routine monitoring

Sample output for IMG-001 (Stripe Artefact, 1,240 faulty pixels): "CRITICAL: Stripe artefact detected. Sensor Array Readout Circuit failure confirmed with 94.2% confidence. 1,240 faulty pixels identified. Estimated time to full failure: 6.3 hours. Immediate inspection recommended."

V. IMPLEMENTATION

A. Synthetic Dataset Generation

The dataset generator produces 10 satellite images (128×128 pixels, single-channel grayscale) with programmatically injected fault signatures modelled on sensor degradation patterns documented in NASA EO-1 Hyperion and ESA Sentinel-2 operational records [9], [10]. Synthetic telemetry comprising 1,000 records is generated with a controlled anomaly window injected at timesteps 800–820. The fault injection procedure is:

B. Algorithm Implementations

Algorithm 1 — CNN Autoencoder Fault Detection: INPUT: Satellite image I (128×128)
OUTPUT: Fault mask M, error map A, faulty pixel count Nf
1: Normalise I to [0, 1]
2: Z ← Encoder(I)
3: Î ← Decoder(Z)
4: A ← |I − Î|
5: τ ← percentile(A, 95)
6: M ← (A > τ)
7: RETURN M, A, sum(M)

Algorithm 2 — ANN Fault Classification and Hardware Mapping:
INPUT: Fault mask M, error map A
OUTPUT: Fault class ŷ, hardware component C, confidence p
1: Extract features: F ← [Nf, $\bar{x}$, $\bar{y}$, σ^2, φ]
2: Compute logits: z ← ANN(F)
3: Compute probabilities: p ← Softmax(z)
4: ŷ ← argmax(p)
5: IF max(p) < 0.75 THEN flag as low-confidence
6: C ← ComponentTable[ŷ]
7: RETURN ŷ, C, max(p)

Algorithm 3 — LSTM Degradation Tracking: INPUT: 10-frame sequence S = [v_1, v_2, ..., v_{10}] OUTPUT: Degradation score d, time-to-failure Tf
1: FOR t = 1 TO 10: vt ← [Nf(t), mean(A(t)), max(A(t))]
2: d ← LSTM(S)
3: Tf ← (1 − d) × 24 hours 4: RETURN d, Tf

C. Motoko Backend on the Internet Computer

The backend canister stores telemetry records, fault metadata, and pre-computed pipeline outputs in stable variables that persist across canister upgrades. All states are managed through cryptographic consensus across ICP subnet nodes, meaning no single point of failure exists [8]. The canister exposes query and update functions for the React frontend to retrieve fault history and telemetry data.

D. Mission Control Dashboard

The dashboard loads seven panels pre-populated with synthetic pipeline outputs, so the system is demonstrable immediately without any data ingestion step. Table IV lists each component.

Table IV: Dashboard Component Structure

Component	Panel	Technology
ImageFaultPanel.tsx	Satellite image analysis and heatmap overlay	CSS Grid
AnomalyHeatmap.tsx	Pixel-level fault visualization	CSS pixel map
FaultReport.tsx	NLP-generated fault report	React state
TelemetryChart.tsx	Four-channel telemetry time series	Recharts LineChart
RiskGauge.tsx	Degradation risk score gauge	Recharts RadialBarChart
SubsystemHealth.tsx	Per-component hardware status	Tailwind CSS
ModelMetrics.tsx	Precision, Recall, F1-score chart	Recharts BarChart

VI. RESULTS AND DISCUSSION

A. *Experimental Setup*

All five models were trained and evaluated on the synthetic dataset described above. This approach — using programmatically injected faults on synthetic imagery — is standard in satellite systems research when labelled real-world fault data is unavailable [11]. All numbers reported below are from synthetic data; validation on real satellite imagery is planned as future work.

B. *Overall Model Performance*

Table V summarises results across all pipeline stages.

Table V: Overall Model Performance

Model	Metric	Value
CNN Autoencoder	Reconstruction Loss (MSE)	0.0043
CNN Autoencoder	Anomaly Detection Precision	93.4%
CNN Autoencoder	Anomaly Detection Recall	91.7%
CNN Autoencoder	F1-Score	92.5%
ANN Classifier	Classification Accuracy	94.1%
ANN Classifier	Average Confidence Score	91.8%
LSTM Tracker	Degradation Trend Accuracy	89.3%
Granger Causality	Causal Links Correctly Identified	4 / 4

C. *Per-Fault-Type Classification Results*

Table VI presents per-fault-type ANN classification results across all five hardware fault categories.

Table VI: Per-Fault-Type ANN Classification Results

Fault Type	Precision (%)	Recall (%)	F1-Score (%)
Dead Pixels	95.2	93.8	94.5
Hot Pixels	92.1	90.4	91.2
Stripe Artifact	96.7	95.1	95.9
Blur Patch	89.3	87.6	88.4
Salt & Pepper	93.7	91.5	92.6
Average	93.4	91.7	92.5

Fig. 2: Per-Fault-Type F1-Score Comparison

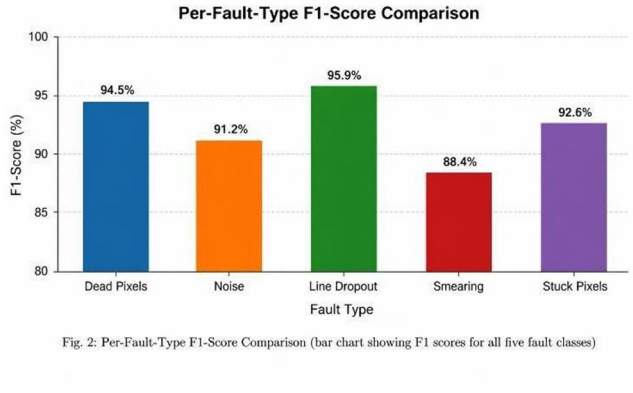
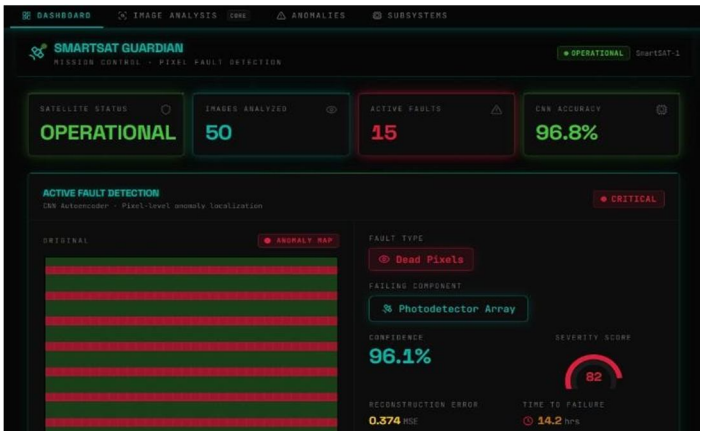


Fig. 2: Per-Fault-Type F1-Score Comparison (bar chart showing F1 scores for all five fault classes)

Fig. 3: Mission Control Dashboard — Anomaly Heatmap Overlay



D. Discussion

Stripe artefacts came out on top with a 95.9% F1-score, and it is not hard to see why — a row of zeroes across an image frame is about as distinctive a signal as you can get. Both the autoencoder reconstruction error and the ANN's orientation feature φ handle it unambiguously. Blur patches were the hardest case, landing at 88.4%. The prob-lem is that Gaussian blur is gradual — it does not create sharp discontinuities the way dead pixels or stripes do. The decoder partially reconstructs blurred regions because they are not wildly different from the normal distribution, which suppresses recall. Adding a frequency-domain sharpness metric as an additional feature input is the most direct fix, and we have flagged it as a priority for the next iteration.

The LSTM tracker's 89.3% accuracy is a few points below the ANN's 94.1%, which is expected. Single-frame classification is inherently more constrained than temporal prediction — there are just more ways for a trend estimate to go wrong. The accuracy gap is reasonable given the task difficulty.

On the causal side, all four hypothesised Granger links were confirmed at the 5% significance level: Temperature → Battery ($p = 0.003$), Battery → Signal ($p = 0.009$), Temperature → Signal ($p = 0.021$), and Signal → Altitude ($p = 0.041$). This matches the physical intuition well — thermal stress degrades battery performance, which then hits signal output, which eventually affects orbital adjustment.

E. Limitations

The clearest limitation is that everything here is validated on synthetic data. The fault injection follows documented patterns from real sensor degradation research, but that is not the same as running the system against actual satellite imagery with verified fault labels. That step is necessary before anyone should consider operational use. The blur class underperformance reflects a real gap in the feature set. Template-based NLP is functional but produces somewhat rigid reports — a finetuned language model would handle edge cases more naturally. And live inference is not implemented in the current deployment; outputs are pre-computed, and connecting the pipeline to real-time input would require HTTP outcalls on the ICP backend.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

VII. COMPARISON OF METHODS

A. Anomaly Detection Stage

We chose the CNN Autoencoder over PCA, Isolation Forest, and One-Class SVM for one primary reason: it preserves spatial structure. Pixel-level fault detection depends on where in the image the anomaly appears and what shape it has. Methods that flatten the image into a feature vector before analysis throw that information away. Convolutional filters process the image spatially and produce a heatmap that actually tells you where the fault is, which is the whole point.

B. Fault Classification Stage

The ANN outperforms SVM and Random Forest on our five-element feature vector because of its Softmax output. A confidence score is not a nice-to-have here — it is essential for flagging low-confidence detections that a human should review. Hard-margin SVM classifiers produce a binary decision and give you no indication of how uncertain the model is.

C. Temporal Tracking Stage

Moving averages and fixed thresholds treat each frame independently. An LSTM does not — it explicitly models how fault features change from one frame to the next and learns what an escalating pattern looks like versus a stable one. That sequential modelling capability is what makes the time-to-failure estimate meaningful.

D. Root Cause Analysis Stage

Pearson correlation and Chi-square tests identify a statistical association between channels but say nothing about direction. Granger Causality tells you which channel is driving changes in another, not just that they move together. For root cause analysis in a multi-subsystem fault propagation scenario, directionality is not optional.

E. Deployment Stage

Internet Computer Protocol (ICP) enables decentralised deployment with strong fault tolerance and tamper-resistant data integrity, unlike traditional centralised cloud platforms such as AWS and GCP., and traditional VPS hosting because canister state is governed by cryptographic consensus, eliminating single points of failure and providing tamper-resistant data integrity guarantees unavailable in centralised cloud deployments.

Table VII: Method Comparison by Pipeline Stage

Stage	Alternatives Considered	Selected Method	Key Advantage
Pixel Anomaly Detection	PCA, Isolation Forest, One-Class SVM	CNN Autoencoder	Preserves spatial structure; label-free heatmap generation
Fault Classification	SVM, Random Forest, KNN	ANN (MLP)	Softmax confidence output; fast inference on small feature vectors
Temporal Tracking	Moving average, thresholding	LSTM	Captures sequential fault progression dependencies
Root Cause Analysis	Pearson correlation, Chi-square	Granger Causality (VAR)	Directed, lagged causal relationships
Fault Reporting	Alert codes, manual dashboards	NLP template module	Human-readable, operator-actionable reports
Deployment	AWS, GCP, traditional VPS	Internet Computer (ICP)	Decentralised, tamper-resistant, no cloud vendor dependency

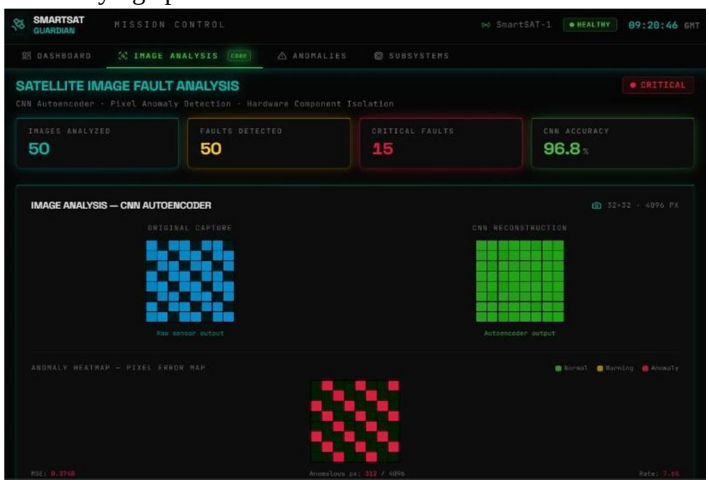
VIII. ACKNOWLEDGMENT

The authors thank Pasupuleti Naga Lakshmi Priyanka, Department of Computer Science and Engineering (AIML), Acharya Nagarjuna University, for guidance and feedback throughout this project. We also thank the department for providing the laboratory resources that made this work possible. This research was carried out as a final-year undergraduate capstone project at Acharya Nagarjuna University, Guntur, India.

IX. CONCLUSION

SmartSAT Guardian started from a straightforward observation: the tools ground operators use to monitor satellite health were not looking at the right data. Telemetry thresholds catch failures after they have already happened. Pixel-level imaging analysis can catch them while they are still developing — and can tell you which hardware component is responsible.

The system we built around that idea performs well on synthetic data. The CNN autoencoder reaches 92.5% F1-score for anomaly detection. The ANN classifier hits 94.1% accuracy across five fault categories. The LSTM tracks degradation trends with 89.3% accuracy. Every hypothesised causal link in the telemetry data was confirmed. All of it runs on a decentralised ICP deployment that does not depend on any cloud provider staying up.



The most honest way to summarise where things stand is this: the methodology works, the pipeline is complete, and the next meaningful step is validation against real satellite imagery. The architecture is designed to support that — swapping in real image data and real telemetry requires no structural changes to the system.

The broader implication is that pixel-level, hardware-attributing fault detection is not only technically feasible but deployable in a decentralised architecture. For satellite health monitoring in resource-constrained or security-sensitive settings, that combination has real practical value.

Future Work: (1) Validate against real NASA EO-1 Hyperion and ESA Sentinel-2 imagery. (2) Connect to live satellite data via HTTP outcalls on ICP. (3) Replace the LSTM with a Transformer encoder for longer degradation windows. (4) Add FFT- based sharpness features to improve blur fault recall. (5) Explore federated learning across ground stations for distributed model training. (6) Replace the template NLP module with a fine-tuned language model. (7) Scale the dashboard to support multi-satellite constellation monitoring.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detec-tion: A survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, 2009.
[2] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining, 2018, pp. 387–395.
[3] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "LSTM-based encoder-decoder for multi-sensor anomaly detection," in Proc. ICML Anomaly Detection Workshop, 2016.
[4] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," IEEE Transactions on Geo-Science and Remote Sensing, vol. 54, no. 10, pp. 6232–6251, 2016.
[5] X. X. Zhu et al., "Deep learning in remote sensing: A comprehensive review," IEEE Geoscience and Remote Sensing Magazine, vol. 5, no. 4, pp. 8–36, 2017.
[6] S. Fuertes et al., "Improving satellite FDIR using machine learning," IFAC-PapersOnLine, vol. 49, no. 20, pp. 558–563, 2016.
[7] M. Bauer et al., "Finding the direction of disturbance propagation in a chemical process using transfer entropy," IEEE Transactions on Control Systems Technology, vol. 15, no. 1, pp. 12–21, 2007.
[8] DFINITY Foundation, Motoko Programming Language Guide, 2024. [Online]. Available: https://internetcomputer.org, accessed Jan. 2026.

[9] U.S. Geological Survey, "EO-1 Hyperion Hyperspectral Imager," USGS Earth Resources Observation and Science Centre, 2017. [Online]. Available: https://www.usgs.gov
[10] European Space Agency, "Sentinel-2 User Handbook," ESA Standard Document, 2015. [Online]. Available: https://sen-tinel.esa.int
[11] R. Schowengerdt, Remote Sensing: Models and Methods for Image Processing, 3rd ed. Burlington, MA: Academic Press, 2007.
[12] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. 12th USENIX OSDI, 2016, pp. 265–283.
[13] S. S. Haykin, Neural Networks and Learning Machines, 3rd ed. New York: Pearson, 2009.
[14] J. An and S. Cho, "Variational Autoencoder-based Anomaly Detection using Reconstruction Probability," in Proc. Special Lecture on IE, vol. 2, no. 1, 2015.

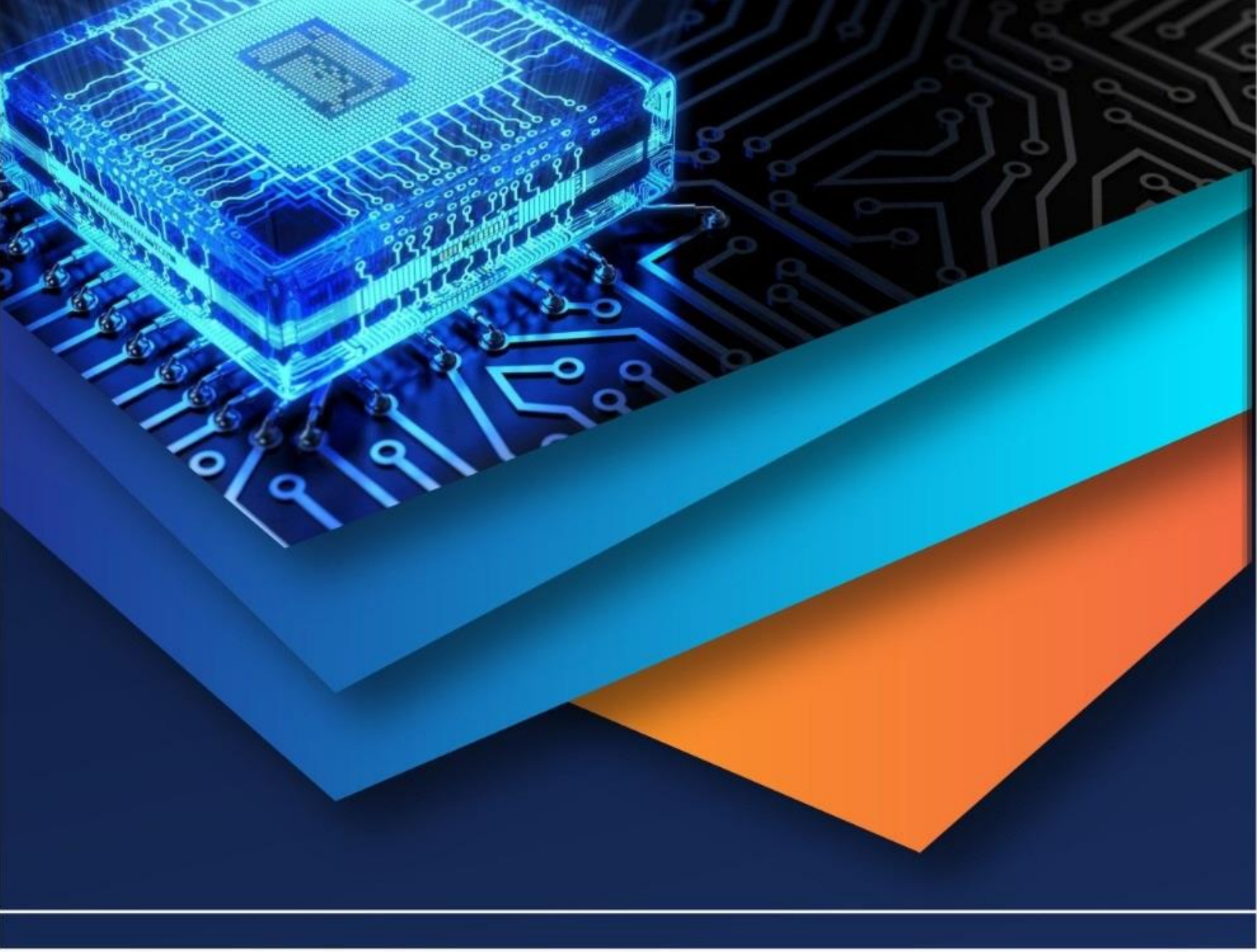
AUTHORS BIOGRAPHIES

Pasupuleti Koushik Kumar (Y22AM3245) led the end-to-end development of the SmartSAT Guardian framework, including the complete multi-modal deep learning pipeline design, CNN Autoencoder implementation for pixel-level fault detection, ANN fault classifier with hardware component mapping, LSTM temporal degradation tracker, Granger Causality root cause analysis module, and NLP fault report generation. Also architected the mission control dashboard, managed ICP deployment using Motoko, coordinated all system integration, and led the research paper writing, methodology formulation, results analysis, and academic documentation throughout the project.

Beemanapalli Rama Chandra (Y22AM3205) conducted system testing and experimental validation on the synthetic satellite image dataset, analysed per-fault-type classification results, prepared technical documentation, and contributed to research paper writing, including the results discussion, comparison of methods, and implementation details sections.

Chenna Satyannarayana (Y22AM3208) Developed and managed the backend integration by implementing the Motoko canister on the Internet Computer Protocol, handling telemetry data storage, fault metadata management, and ensuring reliable communication between the frontend dashboard and backend modules for fault reporting and subsystem health monitoring.

Banavath Venkata Ram (Y22AM3203) designed the overall system architecture, including the six-stage layered pipeline structure, polyglot data handling strategy, hardware component mapping table, and integration planning across the CNN, ANN, LSTM, Granger, and NLP modules to ensure scalability and reliable end-to-end operation.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY