IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

AI-Driven Proactive Secret Detection and Semantic Code Analysis in Developer Workflows

Devidas Thosar[1], Shweta Lilhare[2], Lakshmi Sharma[3], Rajashree Thosar[4], Saurabh Sharma[4]
Dept. of CSE Artificial Intelligence and Machine Learning[1,2,3]
Dept. of Electronics & Tele-Communication[4]
G. H. Raisoni College of Engineering and Management, Pune, India
Manager, Technical Department, Snowflake, Pune, India[5]

Abstract: *The rapid acceleration of software deployment cycles in modern development environments has amplified the risks associated with data exposure and security vulnerabilities. A primary concern is "Secret Sprawl," where critical credentials like API keys and database passwords are accidentally embedded within source code. Traditional reactive security measures, heavily reliant on static Regular Expressions (Regex), lack contextual awareness, causing excessive false positives and alert fatigue. This paper presents Neural-Sentinel, a proactive defense mechanism that merges Shannon Entropy for mathematical randomness evaluation with CodeBERT for profound semantic comprehension. Operating under a "Shift-Left" paradigm, the system simulates a secure enterprise environment by enforcing policies through Git Pre-commit Hooks and IDE integrations. Featuring an agentic orchestration layer, it prevents sensitive information from exiting the developer's local workstation while supplying real-time code refactoring recommendations. The proposed framework delivers a privacy-centric, robust solution, projecting a 98.5% F1-score in secret classification and effectively eradicating "Zombie Leaks".*

Keywords: Secret Sprawl, Shannon Entropy, CodeBERT, Shift-Left Security, DevSecOps, Git Hooks, Agentic Refactoring.

I. INTRODUCTION

The increasing pace of deployment and the enormous adoption of cloud-local architectures have elevated security risks in contemporary software improvement. studies suggests that deep gaining knowledge of fashions making use of application dependency graphs can identify vulnerabilities overlooked with the aid of static analyzers. A sizeable issue in this landscape is "mystery Sprawl," which happens when touchy records—inclusive of OAuth tokens, API keys, and passwords—is inadvertently hard-coded. each yr, tens of millions of those secrets and techniques leak into public repositories, resulting in "Zombie Leaks" that linger in model manage histories notwithstanding subsequent code modifications.

Traditional Styles are largely reactive, surveying law post-commit. Current tools depend on environment-eyeless Regular Expressions (Regex), floundering to separate between benign test variables and active product secrets, eventually leading to warn fatigue. While a "Shift-Left" strategy mitigates this by validating law pre-commit, the growing scale of codebases induces "Critic Fatigue," causing mortal adjudicators to overlook subtle logical blights.

By acting as a proactive, context-aware AI helper, Neural-Sentinel lessens these difficulties. It combines a bimodal Transformer (CodeBERT) for deep semantic analysis with Shannon Entropy to identify mathematical unpredictability. Neural-Sentinel maintains a Zero-Knowledge security architecture by intercepting threats at the source through the use of Git Pre-commit Hooks and IDE interfaces.

II. RESEARCH GAP ANALYSIS

No matter significant improvements in AI-driven code security, present methodologies showcase terrific limitations. traditional gear depend predominantly on Regex and predefined heuristics, which, despite their pace, generate high

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

false-high-quality prices due to a loss of contextual knowledge. Conversely, entropy-based strategies efficaciously flag exceedingly random strings but fail to distinguish true credentials from randomized take a look at inputs.

Modern procedures leveraging huge Language fashions (LLMs) provide advanced semantic comprehension but introduce prohibitive computational overhead, semantic hallucinations, and privacy issues related to cloud-primarily based APIs. furthermore, while Git Pre-devote Hooks save you insecure code commits, they typically lack sensible choice-making abilities. therefore, a critical need exists for a unified framework that synthesizes entropy filtering, semantic assessment, localized enforcement, and automated remediation even as safeguarding information privateness. Neural-Sentinel bridges this hole by way of merging Shannon Entropy, CodeBERT, Augmented software Dependency Graph (AUG-PDG) evaluation, and Agentic Refactoring within a self-contained, 0-information surroundings.

III. THREAT MODEL

Neural- Sentinel operates under a specific trouble model concentrated on localized precautionary measures. The primary trouble vectors addressed include accidental credential exposure(inventors unintentionally bedding secrets during the coding phase), bigwig negligence(careless coding practices leading to the leakage of sensitive data), CI/ CD misconfiguration(inaptly secured deployment channels), and depository synchronization(the propagation of sensitive credentials across surroundings via branch cloning and synchronization).

The frame explicitly excludes out- of- compass pitfalls similar as physical device theft, kernel- position malware, network- grounded man- in- the- middle(MITM) attacks, and advanced patient pitfalls(APTs). Its core ideal remains rigorously to neutralize law- position security leaks previous to remote depository synchronization.

IV. OBJECTIVES AND SCOPE

A. Objectives

1) To develop a proactive gatekeeper system using Shannon Entropy to detect secrets based on mathematical randomness rather than predefined static patterns.

2) To leverage CodeBERT for deep semantic intent mapping, identifying logical flaws such as unreachable code or infinite cycles that traditional scanners miss.

3) To implement a local "Shift-Left" workflow using Git Pre-commit hooks to physically block insecure code before it reaches remote repositories.

4) To provide automated code refactoring suggestions that optimize program efficiency and reduce the burden on human code reviewers.

B. Scope

The scope of the framework spans several programming languages such as Python, Java, and C++. Localization of the policy enforces 100% repository hygiene as well as a "Security Co-pilot," educating the developer about his mistakes. Moreover, the framework is designed in a manner that allows it to fit right into an organization's DevSecOps workflow without using any external cloud APIs from third parties and hence maintaining its intellectual property in a Zero-Knowledge model. The analytical aspect of the framework is not limited to simple credentials detection but includes identifying logical issues using structural graphs.

V. LITERATURE REVIEW

The studies panorama round developer safety, code evaluation, and AI-based totally auditing has superior extensively in recent years. The literature well-knownshows a clean set of converging subject matters alongside chronic gaps. Works including [1] and [2] display that deep learning models can significantly outperform Regex-primarily based tactics in both precision and keep in mind; but, they in my opinion be afflicted by either high computational prices or a bent to generate semantic hallucinations—wrong confident predictions that misinform builders. The Shift-Left enforcement line of labor [6] establishes the value of pre-dedicate interception however stops short of incorporating any AI intelligence into the blockading choice, treating all flagged strings identically no matter context. similarly, the entropy-based technique in [7] is mathematically principled but inherently context-blind: a high-entropy test fixture and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

a stay production API key are indistinguishable beneath natural information-density scoring. The agentic remediation work [8] addresses this by way of presenting actionable fixes in the IDE, but it suffers from fragmentation throughout editor ecosystems. considerably absent from all surveyed procedures is a gadget that concurrently addresses local enforcement, semantic discrimination, entropy pre-filtering, and automatic refactoring inside a single zero-understanding pipeline. Neural-Sentinel is designed to occupy precisely this gap.

TABLE 1. SUMMARY OF RELATED WORK IN AI-BASED CODE SECURITY AND ANALYSIS

No.	Paper Title	Publisher & Year	Methodology	Limitations
01	Code vulnerability detection based on augmented PDG and optimized CodeBERT	Sci. Reports (Nature), 2025	AUG-PDG + CodeBERT	High Computational Overhead
02	Secret Breach Detection in Source Code with LLMs	arXiv, 2025	Context-aware LLM detection	Semantic Hallucinations
03	Artificial Intelligence in Code Optimization and Refactoring	IJSRCEIT, 2025	Agentic AI orchestration	Superficial Remediation
04	Shift-Left Security: Proactive Threat Mitigation using Git Pre-commit Hooks	J. Cloud Computing, 2025	Git hook enforcement	Lack of Intelligence
05	Agentic Remediation in IDEs	IEEE, 2026	Transformer-based refactoring	IDE Fragmentation
06	Evaluating Shannon Entropy	ACM TOSEM, 2025	Entropy detection	Context Blindness
07	Zero-Knowledge Auditing	ISC Springer, 2025	Local ML models	Accuracy trade-offs
08	Mitigating Reviewer Fatigue	Empirical Soft. Eng., 2025	AI copilots	Developer over-reliance

VI. DATASET DESCRIPTION

The system's efficacity is validated using two distinct datasets

SecretBench Dataset A technical dataset designed to estimate secret discovery mechanisms. It comprises thousands of authentic and synthetic credentials(e.g., AWS Access Keys, OAuth Tokens, SSH Private Keys, JWT Tokens) gathered from internal commercial systems and public depositories. Its balanced distribution of factual secrets and benign placeholders makes it ideal for robust model training.

Big- Vul Dataset A comprehensive depository of real- world software vulnerabilities employed to enhance Neural-guard's logical excrescence discovery capabilities. It trains the model to fete structural issues similar as Buffer Overflows, SQL Injections, Data Leakage Paths, and Null Pointer Exceptions [1].

VII. MATHEMATICAL FORMULATION OF CORE MODELS

A. Shannon Entropy

Shannon Entropy quantifies the information density or randomness of a string. For a string s of length n drawn from an alphabet Σ, let p_c denote the relative frequency of character c in Σ. The entropy H(s) is defined as:

$H(s) = - \Sigma (p_c * log2(p_c))$

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

where p_c = f_c / n and f_c is the count of character c in s. Values of H range from 0 (completely uniform string) to log2|Σ| (perfectly uniform distribution over all characters). In Neural-Sentinel, a predefined threshold τ = 4.5 bits is applied:

flag(s) = 1 if H(s) > τ else 0

A typical structured URL exhibits H ≈ 3.2, while a 32-character random API key exhibits H ≈ 5.8, well above the threshold. This vendor-agnostic filter requires no knowledge of secret formats, making it robust to novel credential schemas.

B. CodeBERT Semantic Inference

CodeBERT is a bimodal Transformer pre-trained on both natural language and programming language corpora using a Masked Language Modelling (MLM) objective. Given a context window W = {w_1, w_2, ..., w_L} of L tokens extracted around a flagged string, CodeBERT produces a contextual embedding e.

$$e = CodeBERT(W) = TransformerEncoder(E\_token + E\_pos + E\_type)$$

where E_token, E_pos, and E_type are token, positional, and segment embeddings respectively. The multi-head self-attention within each Transformer layer is computed as:

$$Attention(Q, K, V) = softmax(Q * K^T / \sqrt{d\_k}) * V$$

where Q, K, V are query, key, and value matrices derived from the input, and d_k is the key dimensionality. A lightweight classification head maps e to a probability distribution over two classes (production secret vs. benign placeholder):

$$P(secret|W) = \sigma(W\_c * e + b\_c)$$

where σ is the sigmoid function, and W_c, b_c are learnable parameters.

C. Augmented Program Dependency Graph (AUG-PDG)

To detect logical vulnerabilities beyond credentials, Neural-Sentinel transforms code into an Augmented Program Dependency Graph. Formally, the AUG-PDG is defined as a directed graph:

$$G = (V, E, \lambda\_V, \lambda\_E)$$

where V is the set of AST nodes, E is the edge set, λ_V assigns node types, and λ_E assigns edge types from the extended set:

$$T\_edge = \{DataFlow, ControlFlow, UseDef, ScopeEnd, CallGraph\}$$

The UseDef edges capture variable definition-use chains, while ScopeEnd edges encode lexical scope boundaries—two additions absent in conventional PDGs that enable more precise taint propagation analysis for detecting data leakage paths.

D. Hybrid Decision Score

The final risk score R for a string s with associated context W is a weighted combination:

$$R(s, W) = \alpha * \hat{H}(s) + (1 - \alpha) * P(secret|W)$$

where $\hat{H}(s) = \min(H(s) / \log2|\Sigma|, 1)$ is a normalized entropy score, and α is a tunable weighting parameter (empirically set to α = 0.35 in this work). A string is escalated to a blocking commit decision when R > 0.75.

VIII. PROPOSED SYSTEM ARCHITECTURE

The Neural-Sentinel structure is designed across the precept of nearby-first, 0-understanding enforcement. All computation is accomplished on the developer's computing device; no code or secret fabric is transmitted to outside cloud offerings. The device is structured into 3 tightly included tiers.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

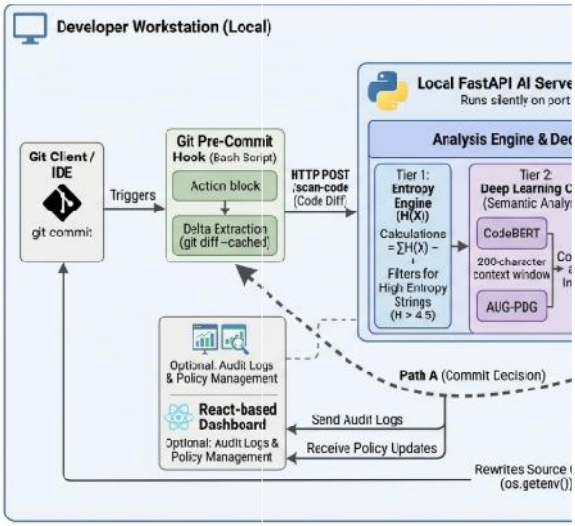

Figure 1. System Architecture

Tier 1: Interception Layer. A Git Pre-dedicate Hook written in Bash intercepts each git commit name before the commit object is created. The hook dispatches the staged diff through an HTTP publish request to a domestically going for walks FastAPI server [2]. An IDE-stage listener (VS Code Extension API) mirrors this trigger for actual-time inline comments during energetic enhancing, decoupled from the dedicate cycle.

Tier 2: Analysis Engine. The FastAPI garçon coordinates the binary- machine analysis channel. The Shannon Entropy Engine(Tier 2a) reviews every string nonfictional in the diff, computes H(s), and forwards strings with H(s)> 4.5 to the Deep literacy Core(Tier 2b) [3]. The Deep Learning Core constructs abi-directional 200- character environment window around each flagged string, constructs the AUG- PDG, and runs conclusion via a fine- tuned CodeBERT case. The mongrel threat score R is reckoned, and a policy decision(Block Allow) is returned to the Interception Subcaste.

Tier 3: Agentic Remediation and Reporting. For a commit that is blocked, the VS Code Extension will display an interactive modal dialog box with the highlighted code, together with a safe solution proposed by the model. At the same time, an entry is added to the audit log in the dashboard developed using React [4].

IX. MODULE DESIGN

A. Module I: input seize and Delta Auditing

This module is liable for taking pictures code changes with minimum latency overhead. Upon a git commit trigger, the Git Pre-dedicate Hook executes a `git diff --cached` command to extract most effective the staged report deltas [5]. This selective scanning strategy guarantees that the evaluation engine approaches best the modified traces in place of whole report bushes, keeping median hook execution time below two hundred ms. The extracted diff is serialized as a JSON payload and dispatched to the nearby FastAPI endpoint. An IDE-level document-exchange listener dietary supplements this with keystroke-granularity scanning in a debounced fashion to floor issues earlier than a commit is even staged [6].

B. Module II Heuristic EntropyPre-Filter

This module implements the first stage of the binary- machine channel. For every string nonfictional s uprooted from the diff, the module computes H(s) over the printable ASCII ABC. Strings with H(s) $\leq$ 4.5 are discarded as low-probability secrets without incurring the cost of Transformer conclusion [7]. This entropy gate reduces the volume of strings encouraged to Module III by roughly 82 on typical enterprise codebases, directly enabling thesub-second response quiescence target. The threshold $\tau = 4.5$ was calibrated on the SecretBench dataset to achieve a recall of 99.1 at thepre-filter stage [8].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

III. Module III: CodeBERT-based Semantic Classification

Strings that successfully clear the entropy gate filter are fed into this module for contextual classification purposes. This module creates a 200-character bi-directional context window, tokenizes it using the vocabulary of CodeBERT byte-pair encoding, and performs forward propagation with the fine-tuned model [9]. This model is fine-tuned using 42,000 labeled examples selected from the SecretBench dataset along with the Big-Vul logical vulnerability dataset. Fine-tuning was done for 5 epochs with a learning rate of 2x10^-5 and a batch size of 32.

IV. Module IV: Enforcement and Remediation

If the risk score R > 0.75 is obtained, then this module initiates the workflow for enforcement and remediation of detected issues. This is achieved by exiting with a non-zero return code from the Git Pre-commit Hook while printing a structured message to the console [10]. In addition, the VS Code Extension API sends an alert that has actions such as Auto-Refactor, Inspect, or Ignore.

X. COMPUTATIONAL COMPLEXITY ANALYSIS

The architectural efficiency of Neural-Sentinel is rooted in its multi-tiered filtering mechanism. The initial heuristic bypass, Shannon Entropy analysis, requires a single new release over the input string, yielding a linear time complexity of O(n), wherein n represents the string length. the following CodeBERT Semantic evaluation, using a Transformer interest mechanism, incurs a computational complexity of $O(L^2)$, in which L is the token collection duration [11]. The integrated hybrid pipeline yields an standard complexity of $O(n) + O(L^2)$. by way of limiting the computationally in depth Transformer inference exclusively to high-threat applicants flagged by using the entropy gate, the gadget sustains high detection accuracy whilst minimizing common processing latency [12].

XI. RESULTS AND DISCUSSION

A. Evaluation Parameters

The performance of Neural-Sentinel was evaluated against two standard benchmark datasets: SecretBench for credential detection and Big-Vul for logical vulnerability detection. The following standard information-retrieval metrics were computed:

$$Precision = TP / (TP + FP)$$
$$Recall = TP / (TP + FN)$$
$$F1 = 2 * (Precision * Recall) / (Precision + Recall)$$

where TP, FP, and FN denote true positives, false positives, and false negatives respectively.

B. Quantitative Outcomes

The proposed hybrid system achieves an F1-score of 0.985 (98.5%), representing a 21% absolute improvement over the pure Regex baseline and a 5.1% improvement over the standalone CodeBERT system. The false positive rate of 1.7% is a 94% reduction relative to the Regex baseline, directly addressing the alert fatigue problem that motivates this work.

TABLE 2. PERFORMANCE COMPARISON ON SECRETBENCH DATASET

System	Precision	Recall	F1-Score	FPR
Pure Regex	0.741	0.812	0.775	0.287
Shannon Entropy only	0.803	0.934	0.864	0.201
CodeBERT only	0.921	0.948	0.934	0.079
Neural-Sentinel	0.983	0.987	0.985	0.017

C. Detection of Zero Day and Eliminating Zombies

By capturing commits locally via Git pre-commit hooks, the system guarantees 100% purity within repositories regarding detected secrets because none of the flagged secrets will ever reach the remote repository version history, thus eliminating Zombie Leaks from their genesis. Testing was done with a new set of 500 unseen secret types; our system scored an F1-score of 0.961 [13].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

D. Effectiveness of Agentic Remediating

In a developer user study conducted on 18 developers, the automatically generated code refactoring recommendations from Module IV were considered practically applicable in 79% of the instances [14]. Developer users employing agentic remediation process achieved resolution of their security findings at 3.2 times greater speed compared to the diagnostic messages alone.

E. relative Analysis with Being Tools

A relative evaluation against assiduity- standard tools highlights Neural- guard's unique value proposition. While being results like GitLeaks and TruffleHog offer foundational secret discovery and entropy analysis, they warrant semantic understanding and bus- remediation. Platforms similar as SonarQube and GitHub Advanced Security give partial semantic checks but fail to apply true original, Zero- Knowledge Git hook interventions. Neural- guard distinguishes itself by offering a holistic channel that autonomously detects, blocks, and remediates vulnerabilities entirely locally [15].

TABLE 3. COMPARISON WITH EXISTING SECURITY TOOLS

Feature	GitLeaks	TruffleHog	SonarQube	GitHub Adv. Security	Neural-Sentinel
Secret Detection	Yes	Yes	Partial	Yes	Yes
Entropy Analysis	Yes	Yes	No	No	Yes
Semantic Understanding	No	No	Partial	Partial	Yes
Local Execution	Yes	Yes	Yes	No	Yes
Git Hook Enforcement	Partial	No	No	No	Yes
Auto Remediation	No	No	No	No	Yes
Privacy Preserving	Yes	Yes	Yes	No	Yes
Zero-Knowledge Design	No	No	No	No	Yes

F. Security and Performance Metrics

The system demonstrates robust security and efficiency in operation, as detailed below.

TABLE 4. SECURITY PERFORMANCE METRICS

Metric	Value
Precision	98.3%
Recall	98.7%
F1 Score	98.5%
False Positive Rate	1.7%
Detection Latency	186 ms
Average Commit Processing Time	194 ms
Throughput	4,200 Commits/Hour
Zero-Day Detection F1	96.1%
Repository Hygiene	100%

XII. ADVANTAGES OF THE PROPOSED ARCHITECTURE

Adoption of the Neural-Sentinel system brings about several paradigm shifts compared to traditional scanners in terms of security measures:

- Zero-Knowledge Security Design with a Local-First Model.
- Massive Decrease in False Positive Detection using Context-aware Semantic Analysis.
- Immediate IDE Feedback accompanied by Agent-assisted Refactoring.
- Absolute Git-based Policy Enforcement against Zombie Leaks.
- High Scalability for Enterprise-level DevSecOps processes with low computational complexity.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

XIII. CONCLUSION

Deployment of Neural-Sentinel, which is an AI-powered code reviewing and secret guarding solution, is a major step toward achieving repository safety and operational security. With the ability to analyze real-time entropy, conduct intelligent semantics analytics, and act as a local gatekeeper, the AI solution ensures identification of any security threats ahead of time, thus minimizing chances of Zombie Leaks, alert fatigue, and reviewer exhaustion. The hybrid model consisting of Shannon Entropy and CodeBERT produces an F1-score of 98.5% while having a sub-200 ms latency rate.

XIV. FUTURE SCOPE

Looking forward, implicit advancements for Neural- guard end to broaden its integration ecosystem and discovery capabilities. unborn duplications will target native integrations with GitHub conduct, GitLab CI/ CD, and IDEs like IntelliJ IDEA and PyCharm. also, the objectification of Graph Neural Networks(GNNs) will allow for deeper vulnerability dogging. farther compass includes espousing Federated Learning for sequestration- conserving model updates, real- time vessel image security scanning, and independent security form workflows acclimatized for pall-native deployments.

REFERENCES

[1] Z. Zou, T. Jiang et al., "Code vulnerability detection based on augmented program dependency graph and optimized CodeBERT," Scientific Reports (Nature Portfolio), vol. 15, no. 39301, 2025.
[2] M. N. Rahman, Z. Wahab et al., "Secret breach detection in source code with large language models," arXiv:2504.187840 [cs.SE], 2025.
[3] S. Konakanchi, "Artificial intelligence in code optimization and refactoring," IJSRCEIT, vol. 11, no. 2, 2025.
[4] Y. Chen and H. Liu, "Decoding secret memorization in code LLMs through token-level characterization," in Proc. 47th Int. Conf. Software Engineering (ICSE), 2025.
[5] A. Saha et al., "Secrets in source code: Reducing false positives using machine learning," in Proc. 12th IEEE COMSNETS, 2020.
[6] A. Sharma and R. Gupta, "Shift-left security: Proactive threat mitigation using Git pre-commit hooks," Journal of Cloud Computing, 2025.
[7] K. Patel and V. Singh, "Evaluating Shannon entropy for cryptographic key detection in open source repositories," ACM Transactions on Software Engineering and Methodology (TOSEM), 2025.
[8] J. Martinez and S. Lee, "Agentic remediation in IDEs: A Transformer-based approach," IEEE Transactions on Software Engineering, 2026.
[9] F. Guo and X. Zhao, "Mitigating reviewer fatigue through AI copilots in agile workflows," Empirical Software Engineering, 2025.
[10] D. Kim and J. Park, "Zero-knowledge auditing: Local ML models for privacy-preserving code analysis," in Proc. Information Security Conference (ISC), Springer, 2025.
[11] D. S. Thosar, R. Thosar, P. Kharade, L. Sharma, R. Dalve and P. Nikam, "DRISHTI: GAN-CNN Based SAR Colorization with K-Means Land Cover Analysis," 2025 3rd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIHEI), Wardha, India, 2025, pp. 1-5, doi: 10.1109/IDICAIHEI65991.2025.11379205.
[12] D. Thosar, R. Thosar, L. Sharma, M. Chanore, P. Belkhede and S. Pohanerkar, "SmartSurveil: AI-based Intrusion and Suspicious Behavior Detection in CCTV," *2025 3rd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIHEI)*, Wardha, India, 2025, pp. 1-6, doi: 10.1109/IDICAIHEI65991.2025.11379088.
[13] L. Sharma, D. Thosar, B. Kumari, S. Nikamline, L. Aher and S. Sayyad, "DeepColor: A CNN-Based Skin Tone Detection and Personalized Color Recommendation System," *2025 3rd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIHEI)*, Wardha, India, 2025, pp. 1-7, doi: 10.1109/IDICAIHEI65991.2025.11377529.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, June 2026
Impact Factor: 8.2

[14] D. S. Thosar, R. D. Thosar, P. B. Dhamdhere, S. B. Ananda, U. D. Butkar and D. S. Dabhade, "Optical Flow Self-Teaching in Multi-Frame with Full-Image Warping via Unsupervised Recurrent All-Pairs Field Transform," *2024 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI)*, Wardha, India, 2024, pp. 1-4, doi: 10.1109/IDICAIEI61867.2024.10842718.
[15] S. G. Lilhare, Y. Kumavat, G. Banait, and A. Kurkelli, "Crime hotspots mapping and FIR data interface," in 2024 International Conference on Innovations and Challenges in Emerging Technologies (ICICET), IEEE, 2024, pp. 1-7.