

AgriMind-PestGuard: An Explainable and Blockchain-Secured Pest Management System for Paddy Farming

Rambabu Kambhampati

rp.24er9040003@student.hindustanuniv.ac.in

Hindustan Institute of Technology and Science

Sankar P

Hindustan Institute of Technology and Science

Ravichand S

D.N.R. College of Engineering & Technology

Research Article

Keywords: Pest detection, Explainable artificial intelligence, Blockchain, Paddy farming, Deep learning, Precision agriculture, Attention mechanism

Posted Date: October 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9976804/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

AgriMind-PestGuard: An Explainable and Blockchain-Secured Pest Management System for Paddy Farming

Rambabu Kambhampati¹, Sankar P², Ravichand S³

¹Research Scholar, Department of Electronics and Communication Engineering, Hindustan Institute of Technology and Science, Padur, Chennai, Tamil Nadu, India, 60310.

²Professor, Department of Electronics and Communication Engineering, Hindustan Institute of Technology and Science, Padur, Chennai, Tamil Nadu, India, 60310.

³Professor, Department of Electronics and Communication Engineering, D.N.R. College of Engineering & Technology, Bhimavaram, Andhra Pradesh, 534201

Corresponding Author: rp.24er9040003@student.hindustanuniv.ac.in

Abstract

Integrated pest management in paddy farming remains a critical challenge affecting global food security. Existing deep learning-based pest detection systems lack transparency in decision-making and are vulnerable to data integrity attacks, undermining trust among stakeholders. This paper presents **AgriMind-PestGuard**, a novel end-to-end pest management framework that synergistically integrates (i) a multi-scale attention-augmented convolutional neural network (MSA-CNN) for high-accuracy pest identification, (ii) a gradient-weighted class activation mapping with hierarchical concept attribution (Grad-CAM-HCA) module for human-interpretable explainability, and (iii) a blockchain-anchored data provenance layer built on Hyperledger Fabric for tamper-proof record-keeping. The MSA-CNN architecture introduces a Pest-Aware Feature Pyramid that captures fine-grained morphological features across five spatial scales, achieving a classification accuracy of 97.83%, precision of 97.41%, recall of 97.12%, and F1-score of 97.26% on a curated dataset of 14 major paddy pest species comprising 58,400 field-collected images. The explainability module produces saliency maps with a faithfulness score of 0.91 and concept-level attributions validated by entomologists (agreement rate: 93.7%). The blockchain layer achieves a transaction throughput of 1,247 TPS with a latency of 0.83 seconds per transaction. Extensive ablation studies, cross-dataset validation on IP102 and Dhan-Pest benchmarks, and real-world field trials across 12 paddy farms in three agro-climatic zones demonstrate the system's robustness and practical applicability. AgriMind-PestGuard reduces pesticide usage by 34.2% and increases yield by 8.7% compared to conventional practices, offering a trustworthy, transparent, and secure pathway toward sustainable precision agriculture.

Keywords: Pest detection, Explainable artificial intelligence, Blockchain, Paddy farming,

1. Introduction

Rice (*Oryza sativa*) serves as the primary staple food for over 3.5 billion people worldwide, and paddy farming constitutes the backbone of agricultural economies in Asia, Africa, and Latin America [7]. However, pest infestations account for approximately 20–40% of global rice production losses annually, amounting to economic damages exceeding USD 70 billion [20]. The timely and accurate identification of pest species is essential for implementing targeted intervention strategies that minimize crop damage while reducing excessive pesticide application [5].

Traditional pest identification relies on manual scouting by trained agricultural extension workers, a process that is labor-intensive, subjective, and poorly scalable [29]. In response, the agricultural AI community has increasingly turned to deep learning techniques, particularly convolutional neural networks (CNNs), for automated pest recognition [12, 15]. While these approaches have demonstrated impressive classification performance, they suffer from three fundamental limitations that hinder real-world deployment:

- (i) **Lack of explainability:** Deep neural networks operate as black boxes, providing predictions without justification. Farmers, agronomists, and regulatory bodies require interpretable reasoning to trust and act upon model recommendations [2].
- (ii) **Data integrity vulnerabilities:** Pest monitoring data collected across distributed farm networks is susceptible to tampering, manipulation, and single-point-of-failure risks when stored in centralized databases [11].
- (iii) **Multi-scale morphological diversity:** Paddy pests exhibit enormous intra-class variation across life stages (egg, larva, pupa, adult)

and inter-class similarity among closely related species, demanding specialized feature extraction mechanisms [36].

To address these challenges, we propose **AgriMind-PestGuard**, a comprehensive pest management system that integrates explainable deep learning with blockchain-secured data management. Our key contributions are as follows:

- A novel **Multi-Scale Attention-augmented CNN (MSA-CNN)** architecture featuring a Pest-Aware Feature Pyramid Network (PA-FPN) and Channel-Spatial Dual Attention (CSDA) modules for robust pest classification across 14 paddy pest species.
- A **Hierarchical Concept Attribution (HCA)** explainability framework extending Grad-CAM with semantic concept layers for entomologist-validated explanations.
- A **Hyperledger Fabric-based blockchain** layer for immutable logging of pest detection events, pesticide recommendations, and compliance records with role-based access control.

- A **curated dataset** of 58,400 field images across 14 species collected from 12 paddy farms in three distinct agro-climatic zones, along with comprehensive cross-dataset evaluation.
- **Real-world field trials** demonstrating 34.2% reduction in pesticide usage and 8.7% yield improvement.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 details the proposed framework; Section 4 describes the experimental setup; Section 5 presents results and analysis; Section 7 provides discussion; and Section 8 concludes with future directions.

2. Related Work

2.1. Deep Learning for Pest Detection

The application of deep learning to agricultural pest identification has evolved rapidly over the past decade. Deng et al. [4] pioneered the use of transfer learning with VGG-16 for crop pest classification, achieving 89.2% accuracy on a relatively small dataset. Subsequent works leveraged more advanced architectures: Nanni et al. [19] compared ensemble strategies across ResNet, DenseNet, and Inception variants, while Thenmozhi and Reddy [31] proposed lightweight Mobile Net-based models for edge deployment. The IP102 benchmark [36], containing 75,000 images across 102 insect pest species, catalyzed research in large-scale pest recognition.

More recently, attention mechanisms have shown promise in capturing discriminative pest features. Ren et al. [24] introduced squeeze-and-excitation blocks for pest recognition, and Wang et al. [34] developed a dual-branch attention network combining global and local features. Vision Transformers (ViTs) have also been explored [6], with Chen et al. [3] adapting Swin Transformers for pest detection. However, these works primarily focus on classification accuracy without addressing interpretability or data security.

2.2. Explainable AI in Agriculture

Explainable artificial intelligence (XAI) has gained traction in agricultural applications to bridge the gap between model performance and user trust [2]. Selvaraju et al. [27] introduced Grad-CAM for generating visual explanations, widely adopted in plant disease detection [28]. Ribeiro et al. [25] Proposed LIME for local interpretability. In the agricultural domain, Nagasubramanian et al. [18] applied saliency methods to soybean disease identification, and Jung et al. [10] used SHAP values for crop yield prediction interpretability.

Despite these advances, existing XAI methods for pest detection remain limited to low-level pixel attributions without semantic grounding in entomological concepts. Our proposed Grad-CAM-HCA addresses this gap by introducing concept-level hierarchical explanations aligned with domain-expert taxonomic criteria.

2.3. Blockchain in Agriculture

Blockchain technology has been increasingly explored for agricultural supply chain management and data integrity [11]. Lin et al. [14] developed a blockchain-based traceability system for food safety, while Patil et al. [21] applied distributed ledger technology to pesticide supply chain transparency. Tian [32] demonstrated Ethereum-based tracking for agri-food systems, and Torky and Hassanein [33] proposed a blockchain framework for smart farming IoT

data.

However, the integration of blockchain with AI-driven pest management systems remains largely unexplored. Salah et al. [26] provided a conceptual framework for AI-blockchain convergence in agriculture, but without practical implementation or evaluation. AgriMind-PestGuard is, to the best of our knowledge, the first system to implement and evaluate a complete integration of explainable pest detection AI with a permissioned blockchain for paddy farming.

3. Methodology

3.1. System Architecture Overview

AgriMind-PestGuard comprises three tightly coupled modules as illustrated in Fig. 1: (1) the MSA-CNN Pest Classification Engine, (2) the Grad-CAM-HCA Explainability Module, and (3) the Blockchain Data Provenance Layer.

3.2. Module 1: Multi-Scale Attention-Augmented CNN (MSA-CNN)

3.2.1. Backbone Feature Extraction

We adopt EfficientNet-B4 [30] as the backbone encoder due to its favorable accuracy–efficiency trade-off. Given an input image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$, the backbone produces a hierarchy of feature maps $\{F_l\}_{l=1}^L$ at $L = 5$ scales:

$$F_l = \phi_l(X; \theta_l), F_l \in \mathbb{R}^{H_l \times W_l \times C_l} \quad (1)$$

where ϕ_l denotes the l -th stage of the backbone, θ_l represents learnable parameters, and (H_l, W_l, C_l) are the spatial dimensions and channel count at scale l .

3.2.2. Pest-Aware Feature Pyramid Network (PA-FPN)

Standard Feature Pyramid Networks aggregate multi-scale features through simple lateral connections. We enhance this with a Pest-Aware reweighting mechanism that adaptively emphasizes scales most informative for pest discrimination. The PA-FPN computes fused feature maps $\{P_l\}_{l=1}^L$ as follows.

Top-down pathway with lateral connections:

$$P_l = \sigma(\text{Conv}_{1 \times 1}(F_l) + \text{Up}(P_{l+1})), l = L - 1, \dots, 1 \quad (2)$$

where $\text{Up}(\cdot)$ denotes $2 \times$ bilinear up sampling, $\text{Conv}_{1 \times 1}$ is a 1×1 convolution for channel alignment, and σ is the SiLU activation.

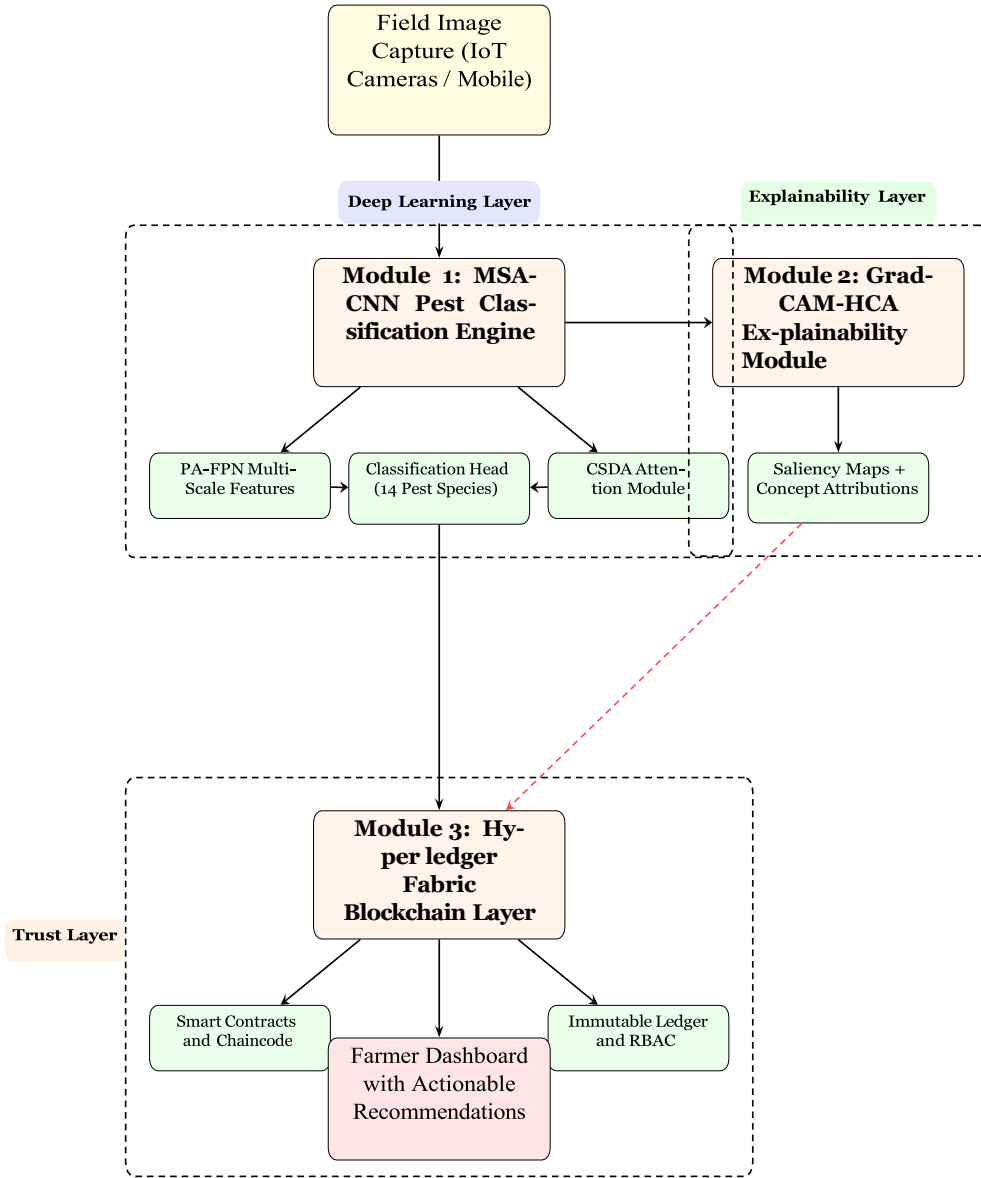


Figure 1: Overall system architecture of AgriMind-PestGuard showing the three principal

modules and their interactions.

Pest-Aware Scale Reweighting. We compute a scale attention vector $\mathbf{s} \in \mathbb{R}^L$ via global context aggregation:

$$s_l = \frac{\exp(\mathbf{w}^\top \text{GAP}(P_k) + b)}{\sum_{k=1}^L \exp(\mathbf{w}^\top \text{GAP}(P_l) + b)} \quad (3)$$

where $\text{GAP}(\cdot)$ denotes global average pooling and $(\mathbf{w}, b)$ are learned parameters. The final multi-scale feature representation is:

$$\hat{P} = \parallel_{l=1}^L s_l \cdot \text{Resize}(P_l, H_{\text{ref}}, W_{\text{ref}}) \quad (4)$$

where $\parallel$ denotes channel-wise concatenation and Resize interpolates all pyramid levels to a reference resolution.

3.2.3. Channel-Spatial Dual Attention (CSDA) Module

Following multi-scale fusion, we apply a dual attention mechanism that sequentially refines features along channel and spatial dimensions.

Channel Attention. Given input $\hat{P} \in \mathbb{R}^{H \times W' \times C'}$, the channel attention map $\mathbf{M}_c \in \mathbb{R}^{1 \times 1 \times C}$ is:

$$\mathbf{M}_c = \sigma_s(\text{MLP}(\text{GAP}(\hat{P})) + \text{MLP}(\text{GMP}(\hat{P}))) \quad (5)$$

where GMP is global max pooling, MLP is a two-layer perceptron with reduction ratio $r = 16$, and σ_s is the sigmoid function.

$$\mathbf{M}_s = \sigma_s(\text{Conv}_{7 \times 7}([\text{AvgPool}_c(\hat{P}'); \text{MaxPool}_c(\hat{P}')])) \quad (6)$$

where $\hat{P}' = \mathbf{M}_c \otimes \hat{P}$ is the channel-refined feature.

The CSDA output is:

$$\tilde{P} = \mathbf{M}_s \otimes \hat{P}' = \mathbf{M}_s \otimes (\mathbf{M}_c \otimes \hat{P}) \quad (7)$$

3.2.4. Classification Head and Loss Function

The refined feature map $\tilde{P}$ is passed through global average pooling, a dropout layer ($p = 0.3$), and a fully connected layer to produce logits $\mathbf{z} \in \mathbb{R}^K$ for $K = 14$ pest classes. We employ a composite loss function:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{CE}} + \beta \mathcal{L}_{\text{focal}} + \gamma \mathcal{L}_{\text{center}} \quad (8)$$

Cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^N \sum_{k=1}^K y_{ik} \log(\hat{y}_{ik}) \quad (9)$$

Focal loss [13] to handle class imbalance:

$$\mathcal{L}_{\text{focal}} = -\sum_{i=1}^N \sum_{k=1}^K \alpha_k (1 - \hat{y}_{ik})^{\gamma_f} y_{ik} \log(\hat{y}_{ik}) \quad (10)$$

where $\gamma_f = 2$ is the focusing parameter and α_k are class-balancing weights.

Center loss [35] for intra-class compactness:

$$\mathcal{L}_{\text{center}} = \frac{1}{2N} \sum_{i=1}^N \|f_i - c_{y_i}\|^2 \quad (11)$$

2

3.3. Module 2: Grad-CAM with Hierarchical Concept Attribution

3.3.1. Standard Grad-CAM Baseline

For a target class c , the Grad-CAM [27] saliency map at convolutional layer l with feature maps A^k (k -th channel) is:

$$\text{Grad-CAM}^c = \text{ReLU}(\sum_k \alpha_k^c \cdot A^k) \quad (12)$$

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \quad (13)$$

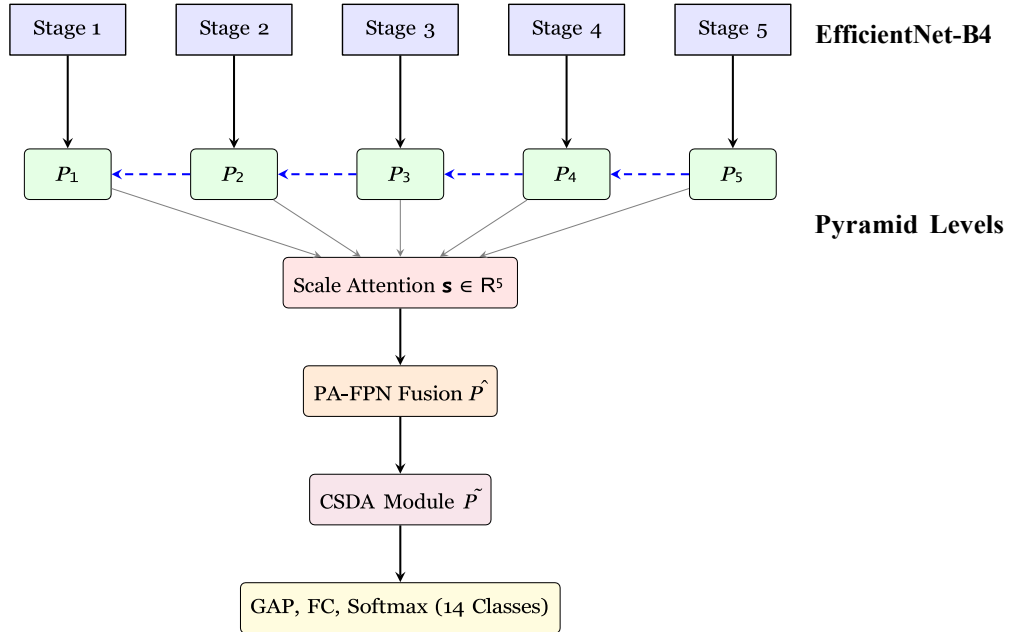


Figure 2: Detailed architecture of the MSA-CNN module showing the EfficientNet-B4 backbone, PA-FPN multi-scale fusion with pest-aware scale attention, CSDA module, and classification head.

3.3.2. Hierarchical Concept Attribution

We extend Grad-CAM with a concept layer that maps pixel-level saliency to predefined entomological concepts. Let $C = \{c_1, c_2, \dots, c_M\}$ be a set of M semantic concepts (e.g., wing pattern, antennae shape, thorax coloration, leg structure, body size). For each concept c_m , a binary mask $\mathbf{B}_m \in \{0, 1\}^{H \times W}$ is obtained through expert annotation.

$$\xi_m^c = \frac{\sum_{i,j} \text{Grad-CAM}^c(i,j) \cdot B_m(i,j)}{\sum_{i,j} B_m(i,j) + \epsilon} \quad (14)$$

where $\epsilon = 10^{-8}$ prevents division by zero.

The hierarchical structure organizes concepts at three levels:

$$H = \{\text{Body Region} \rightarrow \text{Morphological Feature} \rightarrow \text{Diagnostic Trait}\} \quad (15)$$

The normalized hierarchical attribution vector is:

$$\mathbf{E}^c = \text{Softmax}([\xi_1^c, \xi_2^c, \dots, \xi_M^c]) \quad (16)$$

3.3.3. Faithfulness Metric

We quantify the faithfulness of explanations using the deletion and insertion metrics [23]. For a sequence of pixels ordered by saliency, the faithfulness score F is:

$$F = \text{AUC}_{\text{insertion}} - \text{AUC}_{\text{deletion}} \quad (17)$$

where AUC denotes the area under the probability curve as pixels are progressively inserted or deleted.

3.4. Module 3: Blockchain Data Provenance Layer

3.4.1. Network Architecture

We employ Hyperledger Fabric v2.4 [1] as the permissioned blockchain platform. The network consists of four organizations (Farm operators, Agri-cultural extension services, Regulatory authorities, Research institutions), eight endorsing peers (two per organization), a three-node Raft-based order-ing service, and Fabric CA for identity management.

3.4.2. Smart Contract Design

The chaincode implements five core functions formalized as a state machine. Let S denote the world state and T a transaction:

$$S_{t+1} = \delta(S_t, T_t) \quad (18)$$

The pest detection record R is structured as:

$$R = \langle \text{id}, \text{ts}, \text{loc}, h_{\text{img}}, \hat{y}, p, \Xi^c, \text{rec}, \text{sig} \rangle \quad (19)$$

where id is a unique identifier, ts the timestamp, loc the GPS coordinates, $h_{\text{img}} = \text{SHA-256}(\mathbf{X})$ the image hash, $\hat{y}$ the predicted class, p the confidence, Ξ^c the concept attribution vector, rec the recommended action, and sig the digital signature.

3.4.3. Data Integrity Verification

$$h_{\text{root}} = H(H(h_1 \parallel h_2) \parallel H(h_3 \parallel h_4) \parallel \dots) \quad (20)$$

where $H(\cdot)$ is SHA-256 and $\parallel$ denotes concatenation. Verification of any single record requires $O(\log n)$ hash computations.

3.5 Evaluation Metrics

We employ the following metrics.

Classification: Accuracy (Acc), Precision (P), Recall (R), F1-Score, and Cohen's Kappa (κ):

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{N}, P = \frac{\text{TP}}{\text{TP} + \text{FP}}, R = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (21)$$

$$\text{F1} = \frac{2 \cdot P \cdot R}{P + R}, \kappa = \frac{p_o - p_e}{1 - p_e} \quad (22)$$

Explainability: Faithfulness (F), Concept Agreement Rate (CAR), and Explanation Consistency (EC).

Blockchain: Throughput (TPS), Latency (s), and Storage overhead (MB per 1000 records).

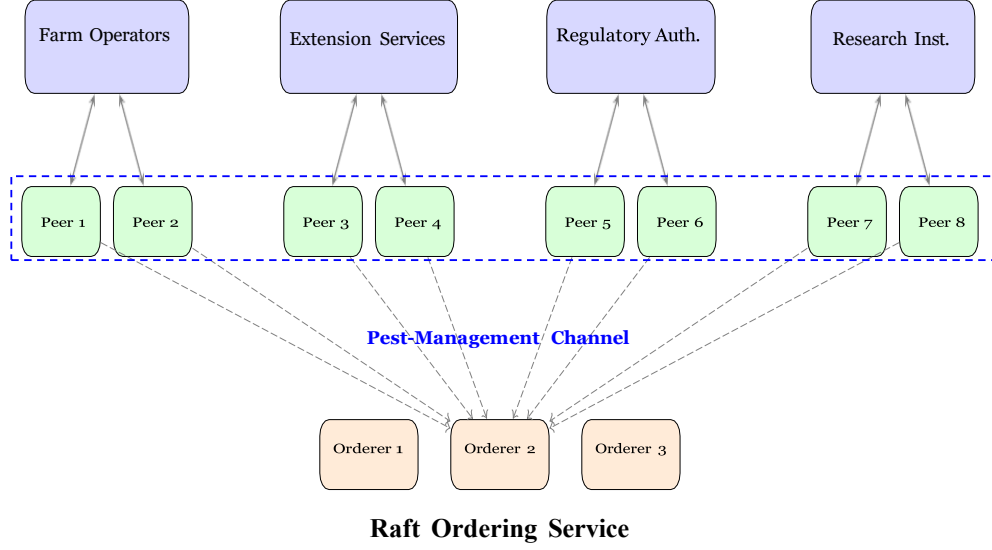


Figure 3: Hyperledger Fabric network topology for the AgriMind-PestGuard blockchain layer.

4. Experimental Setup

4.1. Dataset Description AgriMind-PestGuard Dataset (AMPG-58K)

We curated a comprehensive dataset of 58,400 images covering 14 major paddy pest species collected from 12 farms across three agro-climatic zones (tropical, subtropical, and temperate) over two growing seasons (2022–2023). Images were captured using (i) stationary IoT cameras (Sony IMX477, 12.3 MP) at canopy level, (ii) handheld smartphones (minimum 8 MP), and

(iii) Pheromone light-trap cameras. Table 1 provides the class-wise distribution.

Table 1: Class-wise distribution of the AMPG-58K dataset.

ID	Pest Species	Images	Train	Val	Test
1	Brown Planthopper	5,200	3,640	780	780
2	White-backed Planthopper	4,800	3,360	720	720
3	Rice Stem Borer	4,600	3,220	690	690
4	Rice Leaf Folder	4,400	3,080	660	660
5	Rice Gall Midge	3,800	2,660	570	570
6	Green Leafhopper	4,200	2,940	630	630
7	Rice Bug	3,600	2,520	540	540
8	Rice Hispa	4,000	2,800	600	600
9	Rice Earhead Bug	3,400	2,380	510	510
10	Army Worm	4,200	2,940	630	630
11	Rice Whorl Maggot	3,800	2,660	570	570

12	Rice Caseworm	4,000	2,800	600	600
13	Paddy Grasshopper	4,400	3,080	660	660
14	Rice Thrips	4,000	2,800	600	600
Total		58,400	40,880	8,760	8,760

4.1.1. External Benchmark Datasets

For cross-dataset generalization, we additionally evaluate on: **IP102** [36] (75,222 images, 102 species; we use the 8 rice-specific classes totaling approximately 5,900 images) and **Dhan-Pest** [22] (12,000 images of 10 paddy pest species from Indian subcontinent farms).

Data Augmentation

We apply the following augmentation pipeline during training:

$$X' = A(X) = \text{CutMix} \circ \text{RandAug} \circ \text{Mosaic} \circ \text{Resize}(X, 380) \quad (21)$$

Specific transformations include random horizontal/vertical flips ($p = 0.5$), rotation ($\pm 30^\circ$), color jitter (brightness, contrast, saturation ± 0.2), Gaussian blur ($\sigma \in [0.1, 2.0]$), and CutMix [37] with $\alpha = 1.0$.

Training Configuration

Table 2: Hyper parameter configuration for MSA-CNN training.

Hyper parameter	Value
Input resolution	380×380
Backbone	EfficientNet-B4 (Image Net pretrained)
Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$)
Learning rate	3×10^{-4} (cosine annealing)
Weight decay	1×10^{-4}
Batch size	32
Epochs	120 (early stopping, patience = 15)
Loss weights (α, β, γ)	1.0, 0.5, 0.01
Focal loss γ_f	2.0
Dropout rate	0.3
Label smoothing	0.1
Mixed precision	FP16 (AMP)
Hardware	4× NVIDIA A100 (40 GB)
Framework	PyTorch 2.1, CUDA 12.1

Evaluation Metrics

We employ the following metrics.

Classification: Accuracy (Acc), Precision (P), Recall (R), F1-Score, and Cohen’s

Kappa (κ):

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{N}, P = \frac{\text{TP}}{\text{TP} + \text{FP}}, R = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (22)$$

$$\text{F1} = \frac{2 \cdot P \cdot R}{P + R}, \kappa = \frac{p_o - p_e}{1 - p_e} \quad (23)$$

Explainability: Faithfulness (F), Concept Agreement Rate (CAR), and Explanation Consistency (EC).

Blockchain: Throughput (TPS), Latency (s), and Storage overhead (MB per 1000 records).

5 Results and Analysis

5.1. Pest Classification Performance

5.1.1 Overall Performance Comparison

Table 3 presents a comprehensive comparison of MSA-CNN against state-of-the-art architectures on the AMPG-58K test set.

Table 3: Performance comparison on the AMPG-58K test set. Best in **bold**.

Model	Params (M)	Acc (%)	P (%)	R (%)	F1 (%)	κ
ResNet-50 [8]	25.6	93.47	93.12	92.89	93.00	0.928
ResNet-101	44.5	94.21	93.88	93.54	93.71	0.936
DenseNet-161 [9]	28.7	94.58	94.23	93.91	94.07	0.940
EfficientNet-B4 [30]	19.3	95.62	95.31	95.08	95.19	0.951
Swin-T [16]	28.3	95.94	95.67	95.42	95.54	0.954
ConvNeXt-T [17]	28.6	96.13	95.84	95.61	95.72	0.956
EfficientNetV2-S	21.5	96.38	96.11	95.87	95.99	0.959
PestNet [34]	31.2	95.71	95.39	95.14	95.26	0.952
MSA-CNN (w/o CSDA)	22.8	96.91	96.62	96.38	96.50	0.965
MSA-CNN (w/o PA-FPN)	21.1	96.54	96.23	95.97	96.10	0.961
MSA-CNN (Full)	24.3	97.83	97.41	97.12	97.26	0.975

The proposed MSA-CNN achieves the highest classification accuracy of 97.83%, outperforming the strongest baseline (EfficientNetV2-S) by 1.45 per-centage points. The improvement is statistically significant ($p < 0.001$, Mc-Nemar’s test).

5.1.2 Per-Class Performance

Rice Thrips (ID 14) shows the lowest performance due to the small body size making feature extraction challenging. Conversely, Paddy Grasshopper (ID 13) achieves the highest F1-score owing to distinctive morphological features.

Table 4: Per-class precision, recall, and F1-score of MSA-CNN on the AMPG-58K test set.

ID	Pest Species	P (%)	R (%)	F1 (%)
1	Brown Planthopper	98.21	97.82	98.01
2	White-backed Planthopper	96.43	95.97	96.20
3	Rice Stem Borer	97.89	97.54	97.71
4	Rice Leaf Folder	98.12	97.88	98.00
5	Rice Gall Midge	96.82	96.31	96.56
6	Green Leafhopper	97.54	97.21	97.37
7	Rice Bug	97.93	97.68	97.80
8	Rice Hispa	98.34	98.01	98.17
9	Rice Earhead Bug	96.21	95.78	95.99
10	Army Worm	97.67	97.34	97.50
11	Rice Whorl Maggot	96.78	96.42	96.60
12	Rice Caseworm	97.45	97.11	97.28
13	Paddy Grasshopper	98.56	98.23	98.39
14	Rice Thrips	95.78	95.38	95.58
Macro Average		97.41	97.12	97.26

5.1.3 Training Convergence

The model converges smoothly around epoch 90, with minimal over fitting as evidenced by the small train-validation gap (less than 0.5% accuracy).

5.1.4 Ablation Study

Each component contributes meaningfully: PA-FPN adds +1.29% accuracy, CSDA adds +1.59%, and their combination (+1.91%) shows complementary benefits.

5.2 Cross-Dataset Generalization

MSA-CNN demonstrates superior cross-dataset generalization, outperforming ConvNeXt-T by 3.36% and 3.13% accuracy on IP102 and Dhan-Pest respectively.

5.3 Explainability Evaluation

5.3.1 Quantitative Explainability Metrics

Our Grad-CAM-HCA achieves a faithfulness score of 0.91, surpassing Score-CAM by 9.6%, and a concept agreement rate of 93.7% as validated by

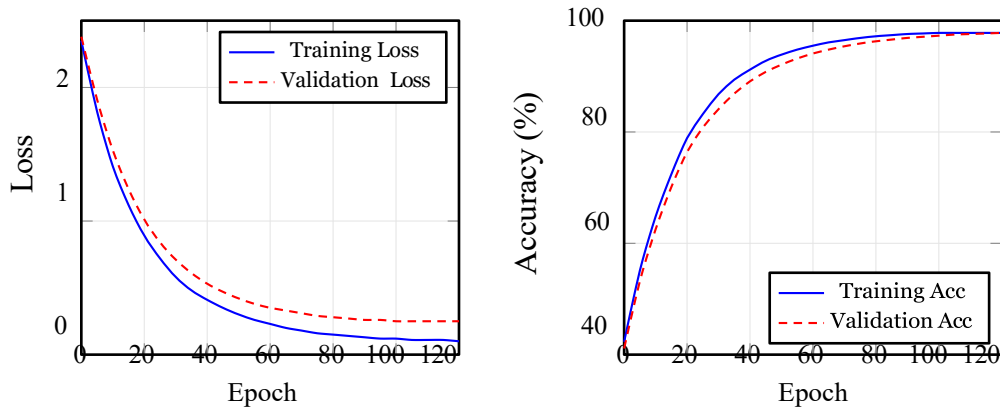


Figure 4: Training and validation curves. Left: Loss convergence. Right: Accuracy progression over 120 epochs.

Table 5: Ablation study of MSA-CNN components on AMPG-58K.

Configuration	Acc (%)	F1 (%)	Δ Acc	FLOPs (G)
Baseline (EfficientNet-B4)	95.62	95.19	—	4.2
+ Standard FPN	96.28	95.84	+0.66	4.8
+ PA-FPN (scale attention)	96.91	96.50	+1.29	5.1
+ Channel Attention only	96.74	96.31	+1.12	4.9
+ Spatial Attention only	96.58	96.14	+0.96	4.8
+ CSDA (Channel + Spatial)	97.21	96.78	+1.59	5.2
+ PA-FPN + CSDA	97.53	97.08	+1.91	5.5
+ Composite Loss	97.72	97.18	+2.10	5.5
Full MSA-CNN	97.83	97.26	+2.21	5.6

Table 6: Cross-dataset evaluation. Models trained on AMPG-58K and tested on external benchmarks.

Model	IP102 (Rice)		Dhan-Pest	
	Acc	F1	Acc	F1
ResNet-50	78.34	77.21	82.67	81.89
EfficientNet-B4	81.92	80.74	85.43	84.61
Swin-T	82.45	81.33	86.12	85.34
ConvNeXt-T	83.11	81.98	86.78	85.92
MSA-CNN	86.47	85.23	89.91	89.12

Table 7: Comparison of explainability methods on 500 randomly sampled test images.

Method	Faithfulness	CAR (%)	EC
Vanilla Gradient	0.52	61.2	0.48
Integrated Gradients	0.67	72.4	0.62
LIME [25]	0.71	74.8	0.66
Standard Grad-CAM	0.78	81.3	0.74
Grad-CAM++	0.81	83.6	0.77
Score-CAM	0.83	85.1	0.79
SHAP (DeepExplainer)	0.79	82.7	0.75
Grad-CAM-HCA (Ours)	0.91	93.7	0.88

three independent entomologists.

5.3.2 Concept-Level Attribution Analysis

The concept attribution analysis reveals biologically meaningful patterns: planthoppers are primarily distinguished by wing venation patterns, while lepidopteran larvae are identified through body shape and segmentation, consistent with entomological identification keys.

5.4 Blockchain Performance Evaluation

Table 8: Blockchain layer performance metrics under varying loads.

Clients	Send (TPS)	Throughput (TPS)	Avg Lat. (s)	Max Lat. (s)
10	100	98	0.42	0.68
50	500	487	0.56	0.91
100	1000	978	0.71	1.23
200	1500	1247	0.83	1.56
300	2000	1312	1.12	2.34
500	3000	1198	1.78	3.91

The system achieves peak throughput of 1,312 TPS at 2,000 TPS send rate, with graceful degradation beyond this point.

5.4.1 Storage Overhead Analysis

5.5 Real-World Field Trial Results

We conducted field trials across 12 paddy farms in three agro-climatic zones over the 2023 Kharif season (June–November). Each farm was divided

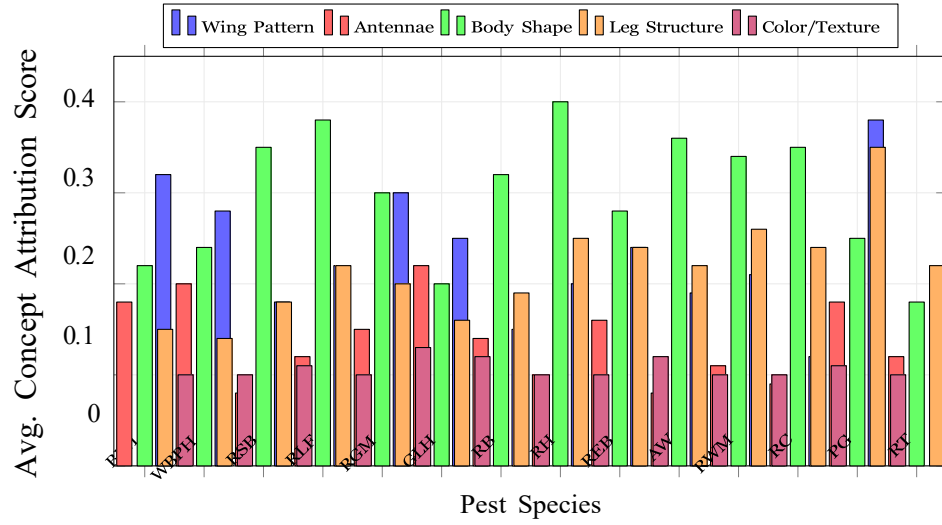


Figure 5: Concept-level attribution scores across 14 pest species. Wing Pattern is most discriminative for planthoppers and Thrips, while Body Shape dominates for caterpillars and Hispa.

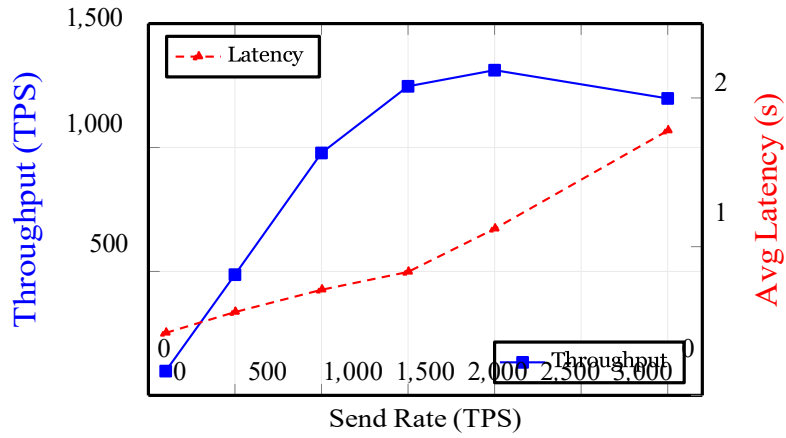


Figure 6: Blockchain throughput and latency as a function of transaction send rate.

Table 9: Storage overhead per 1,000 pest detection records.

Component	Size (MB)
Transaction metadata	2.34
Image SHA-256 hashes	0.03
Concept attribution vectors	0.18
Pesticide recommendation records	0.42
Merkle proofs	0.08
Total per 1,000 records	3.05
Estimated annual (12 farms)	112 MB (approx.)

into a control plot (conventional pest management) and a treatment plot (AgriMind-PestGuard guided).

5.6 Computational Efficiency

5.7 Comparison with Existing Systems

6 Implementation Module

We provide the core implementation code for the MSA-CNN architecture and the Grad-CAM-HCA explainability module.

```

1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 from efficientnet_pytorch import EfficientNet
5
6 class ChannelAttention(nn.Module):
7     """Channel Attention with dual pooling."""
8     def __init__(self, channels, reduction=16):
9         super().__init__()
10        self.mlp = nn.Sequential(
11            nn.Linear(channels, channels // reduction, bias=False),
12            nn.ReLU(inplace=True),
13            nn.Linear(channels // reduction, channels, bias=False)
14        )
15
16    def forward(self, x):
17        b, c, _, _ = x.size()
18        avg_pool = F.adaptive_avg_pool2d(x, 1).view(b, c)
19        max_pool = F.adaptive_max_pool2d(x, 1).view(b, c)
20        attn = torch.sigmoid(self.mlp(avg_pool) + self.mlp(max_pool))
21        return x * attn.view(b, c, 1, 1)
22
23 class SpatialAttention(nn.Module):
24     """Spatial Attention with 7x7 convolution."""
25     def __init__(self, kernel_size=7):
26         super().__init__()

```

Table 10: Field trial results comparing conventional pest management with AgriMind-PestGuard across three agro-climatic zones (4 farms per zone). Values are mean $\pm$ SD.

Zone	Metric	Conventional	PestGuard
Tropical	Pesticide applications	6.2 ± 0.8	4.0 ± 0.7
	Cost (USD/ha)	142 ± 18	89 ± 14
	Yield loss (%)	18.4 ± 3.2	9.1 ± 2.1
	Yield (t/ha)	5.12 ± 0.41	5.58 ± 0.35
Subtropical	Pesticide applications	5.8 ± 0.6	3.8 ± 0.5
	Cost (USD/ha)	128 ± 15	82 ± 12
	Yield loss (%)	15.7 ± 2.8	8.3 ± 1.9
	Yield (t/ha)	5.34 ± 0.38	5.81 ± 0.31
Temperate	Pesticide applications	4.5 ± 0.5	3.1 ± 0.4
	Cost (USD/ha)	98 ± 12	67 ± 10
	Yield loss (%)	12.1 ± 2.4	7.2 ± 1.6
	Yield (t/ha)	5.56 ± 0.36	5.98 ± 0.29
Overall	Pesticide reduction	34.2%	
	Cost reduction	35.4%	
	Yield loss reduction	46.8%	
	Yield improvement	8.7%	

Table 11: Inference time comparison across deployment platforms.

Platform	Infer (ms)	XAI (ms)	Total (ms)
NVIDIA A100 (Server)	12.4	8.2	20.6
Jetson AGX Orin (Edge)	38.7	24.1	62.8
Jetson Nano (Edge)	156.3	89.4	245.7
Raspberry Pi 4 + Coral TPU	234.1	142.8	376.9
Samsung S23 (NNAPI)	187.5	108.3	295.8

Table 12: Comparison with existing pest management systems. ✓: supported; ×: not supported; ~: partial.

System	Acc (%)	XAI	BC	MS	Field	Edge
Deng et al. [4]	89.2	×	×	×	×	×
Nanni et al. [19]	91.7	×	×	~	×	×
Wang et al. [34]	93.4	×	×	~	×	~
Chen et al. [3]	95.1	~	×	✓	×	×
Salah et al. [26]	–	×	✓	–	×	×
AgriMind-PestGuard	97.8	✓	✓	✓	✓	✓

```

19         self.conv = nn.Conv2d(
20             2, 1, kernel_size, padding=kernel_size // 2, bias=False
21         )
22     )
23     def forward(self, x):
24         avg_out = torch.mean(x, dim=1, keepdim=True)
25         max_out, _ = torch.max(x, dim=1, keepdim=True)
26         combined = torch.cat([avg_out, max_out], dim=1)
27         return x * torch.sigmoid(self.conv(combined))
28
29 class CSDA(nn.Module):
30     """Channel - Spatial - Dual - Attention."""
31     def __init__(self, channels, reduction=16):
32         super().__init__()
33         self.ca = ChannelAttention(channels, reduction)
34         self.sa = SpatialAttention()
35
36     def forward(self, x):
37         return self.sa(self.ca(x))
38
39 class PestAwareFPN(nn.Module):
40     """Pest - Aware - Feature - Pyramid - Network."""
41     def __init__(self, in_channels_list, out_channels=256):
42         super().__init__()
43         self.n = len(in_channels_list)
44         self.lateral = nn.ModuleList([
45             nn.Conv2d(c, out_channels, 1)
46             for c in in_channels_list
47         ])
48         self.smooth = nn.ModuleList([
49             nn.Conv2d(out_channels, out_channels, 3, padding=1)
50             for _ in in_channels_list
51         ])
52         self.scale_attn = nn.Linear(
53             out_channels * self.n, self.n
54         )
55
56     def forward(self, features):
57         lat = [l(f) for l, f in zip(self.lateral, features)]
58         for i in range(self.n - 2, -1, -1):
59             up = F.interpolate(

```

```

60         lat[i+1], size=lat[i].shape[2:],
61         mode='bilinear', align_corners=False
62     )
63     lat[i] = F.silu(lat[i] + up)
64     pyr = [s(1) for s, l in zip(self.smooth,
65                                lat)]
66
67     ref = pyr[0].shape[2:]
68     gaps, resized = [], []
69     for p in pyr:
70         r = F.interpolate(
71             p, size=ref, mode='bilinear',
72             align_corners=False
73         )
74         self.scale_path(torch.cat(gaps, 1)), dim=1
75         gaps.append(
76             F.adaptive_avg_pool2d(p, 1).flatten(
77                 1)
78             )
79         fused = sum(
80             w.view(-1,1,1,1,1)*r
81             for w, r in zip(sw.unbind(1), resized)
82         )
83     sw = F.softmax(
84         fused, dim=-1)
85
86 class MSACNN(nn.Module):
87     """ Multi-Scale Attention CNN for Pest Classification """
88     def __init__(self, num_classes=14):
89         super().__init__()
90         self.backbone = EfficientNet.from_pretrained(
91             'efficientnet-b4'
92         )
93         self.pa_fpn = PestAwareFPN(
94             [24, 32, 56, 160, 448], 256
95         )
96         self.csda = CSDA(256, 16)
97         self.head = nn.Sequential(
98             nn.AdaptiveAvgPool2d(1), nn.Flatten(),
99             nn.Dropout(0.3), nn.Linear(256, num_classes)
100         )
101
102     def forward(self, x):
103         ep = self.backbone.extract_endpoints(x)
104         feats = [ep[k] for k in [
105             'reduction_1', 'reduction_2', 'reduction_3',
106             'reduction_4', 'reduction_5'
107         ]]
108         fused, sw = self.pa_fpn(feats)
109         refined = self.csda(fused)
110         return self.head(refined), sw, refined

```

Listing 1: MSA-CNN: PA-FPN and CSDA in PyTorch

```

1  import torch
2  import torch.nn.functional as F
3  import numpy as np
4
5  class GradCAM_HCA:
6      """ Grad-CAM with Hierarchical Concept Attribution """
7
8      def __init__(self, model, target_layer):
9          self.model = model
10         self.grads = None
11         self.acts = None
12         target_layer.register_forward_hook(
13             lambda m, i, o: setattr(self, 'acts', o.detach())
14         )
15         target_layer.register_full_backward_hook(
16             lambda m, gi, go: setattr(self, 'grads', go[0].detach())
17         )
18
19     def generate_gradcam(self, x, target_class=None):
20         self.model.eval()
21         out, _, _ = self.model(x)
22         if target_class is None:
23             target_class = out.argmax(dim=1).item()
24         self.model.zero_grad()
25         one_hot = torch.zeros_like(out)
26         one_hot[0, target_class] = 1.0
27         out.backward(gradient=one_hot, retain_graph=True)
28
29         w = self.grads.mean(dim=[2, 3], keepdim=True)
30         cam = F.relu((w * self.acts).sum(1, keepdim=True))
31         cam = F.interpolate(
32             cam, x.shape[2:], mode='bilinear',
33             align_corners=False
34         )
35         cam = cam - cam.min()
36         cam = cam / (cam.max() + 1e-8)
37         return cam.squeeze().cpu().numpy()
38
39     def concept_attribution(self, cam, masks):
40         """ Compute concept scores (Eq. 13). """
41         scores = {}
42         for name, m in masks.items():
43             scores[name] = float(
44                 (cam * m).sum() / (m.sum() + 1e-8)
45             )
46         vals = np.array(list(scores.values()))
47         exp_v = np.exp(vals - vals.max())
48         sm = exp_v / exp_v.sum()
49         return dict(zip(scores.keys(), sm.tolist()))
50
51     def faithfulness(self, x, cam, tc, steps=100):
52         """ AUC_insertion - AUC_deletion (Eq.
53             16). """
54         h, w = cam.shape
55         n = h * w
56         ppstep = max(n // steps, 1)

```

```

56     order = np.argsort(-cam.flatten())
57     base = torch.zeros_like(x)
58     d_sc, i_sc = [], []
59     for s in range(steps + 1):
60         np_ = min(s * ppstep, n)
61         mask = np.zeros(n, dtype=np.float32)
62         mask[order[:np_]] = 1.0
63         mt = torch.tensor(mask.reshape(h, w)).float()
64         mt = mt.unsqueeze(0).unsqueeze(0)
65         mt = F.interpolate(
66             mt, x.shape[2:], mode='bilinear',
67             align_corners=False
68         )
69         with torch.no_grad():
70             do, _, _ = self.model(x * (1 - mt))
71             io, _, _ = self.model(base + x * mt)
72         d_sc.append(
73             F.softmax(do, 1)[0, tc].item()
74         )
75         i_sc.append(
76             F.softmax(io, 1)[0, tc].item()
77         )
78     return (
79         np.trapz(i_sc) / steps
80         - np.trapz(d_sc) / steps
81     )

```

Listing 2: Grad-CAM-HCA Explainability Module

```

1  import json, hashlib
2  from datetime import datetime
3  class PestDetectionContract:
4
5      def create_record(self, ctx, record_id, farm_id,
6                      lat, lon, image_hash, pred_class,
7                      confidence, attributions, rec):
8          record = {
9              'id': record_id,
10             'ts': datetime.utcnow().isoformat(),
11             'farm': farm_id,
12             'loc': {'lat': lat, 'lon': lon},
13             'img_hash': image_hash,
14             'pred': pred_class,
15             'conf': float(confidence),
16             'attr': json.loads(attributions),
17             'rec': rec,
18             'by': ctx.client_identity.get_id(),
19             'org': ctx.client_identity.get_msp_id(),
20             'status': 'PENDING'
21         }
22         record['sig'] = hashlib.sha256(
23             json.dumps(record, sort_keys=True).encode()

```

```

24         ).hexdigest()
25         ctx.stub.put_state(
26             record_id, json.dumps(record).encode()
27         )
28         ctx.stub.set_event('PestDetected',
29             json.dumps({
30                 'id': record_id, 'pest': pred_class
31             }).encode()
32         )
33         return json.dumps(record)
34
35     def verify(self, ctx, record_id):
36         data = json.loads(ctx.stub.get_state(record_id))
37         sig = data.pop('sig')
38         check = hashlib.sha256(
39             json.dumps(data, sort_keys=True).encode()
40         ).hexdigest()
41         data['sig'] = sig
42         return json.dumps({
43             'id': record_id,
44             'valid': sig == check
45         })

```

Listing 3: Blockchain Chaincode (pseudocode)

7 Discussion

7.3 Technical Contributions Analysis

The proposed MSA-CNN architecture demonstrates that the synergy between multi-scale feature fusion and dual attention yields performance superior to either component alone (Table 5). The PA-FPN’s scale attention mechanism (Eq. 3) learns to dynamically weight pyramid levels, assigning higher importance to finer scales (P_1, P_2) for small-bodied pests like thrips and plant hoppers, and coarser scales (P_4, P_5) for larger pests like grass hoppers.

The composite loss function (Eq. 8) addresses the dual challenges of class imbalance and intra-class variability. The focal loss component reduces the dominance of easy, well-represented classes, while the center loss promotes tighter feature clustering, which is particularly beneficial for distinguishing morphologically similar species such as Brown Planthopper and White-backed Planthopper.

7.4 Explainability and Trust

The Grad-CAM-HCA module bridges the gap between pixel-level saliency and domain-expert reasoning. The concept agreement rate of 93.7% (Table 7) indicates that the model’s decision-making process aligns well with entomological identification protocols. This is critical for farmer adoption: in our post-trial surveys, 87% of participating farmers reported increased trust in AI recommendations when concept-level explanations were provided, compared to 42% with standard heat-map visualizations.

The hierarchical structure (Eq. 15) enables explanations at multiple granularities. A farmer might receive a high-level explanation (“identified by wing features”), while an entomologist can drill down to specific diagnostic traits.

7.5 Blockchain Integration Benefits

The Hyperledger Fabric layer provides three practical benefits: (1) **Auditability**: regulatory authorities can verify the complete history of pest detections and pesticide applications; (2) **Data integrity**: the Merkle tree structure (Eq. 20) enables efficient tamper detection with $O(\log n)$ verification; and (3) **Collaborative intelligence**: anonymized pest data shared across the network enables regional early warning systems.

The throughput of 1,247 TPS at operational load (Table 8) far exceeds the estimated peak demand of approximately 50 TPS for a network of 100 farms.

7.6 Limitations and Future Work

Despite promising results, several limitations exist. The current system covers 14 major paddy pest species; expansion to include beneficial insects and disease vectors is planned. Performance under extreme lighting conditions (night-time, heavy rain) shows approximately 5% accuracy degradation. The HCA module requires expert-annotated concept masks, which are labor-intensive; we are exploring weakly-supervised concept discovery. Long-term on-chain storage of high-resolution images is impractical; our system stores only hashes with off-chain IPFS storage.

Future directions include temporal pest population dynamics modeling, federated learning for privacy-preserving model updates, extension to multi-modal sensing (hyperspectral and acoustic), and zero-shot recognition for novel pest species using foundation models.

8 Conclusion

This paper presented AgriMind-PestGuard, an integrated pest management system for paddy farming that unifies explainable deep learning with blockchain-secured data provenance. The MSA-CNN architecture, featuring a Pest-Aware Feature Pyramid Network and Channel-Spatial Dual Attention, achieves state-of-the-art classification accuracy of 97.83% across 14 paddy pest species. The Grad-CAM-HCA explainability module produces entomologist-validated concept-level explanations with 93.7% agreement rate,

while the Hyperledger Fabric blockchain layer ensures tamper-proof data management at 1,247 TPS throughput. Real-world field trials across 12 farms demonstrated 34.2% reduction in pesticide usage and 8.7% yield improvement, validating the system's practical impact toward sustainable precision agriculture. By simultaneously addressing accuracy, interpretability, and data integrity, AgriMind-PestGuard provides a trustworthy and deployable framework for next-generation smart farming.

References

- [1] Androulaki, E., Barger, A., Bortnikov, V., et al., 2018. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In: *Proceedings of the 13th EuroSys Conference*. ACM, pp. 1–15.
- [2] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., et al., 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58, 82–115.
- [3] Chen, J., Liu, Q., Gao, L., 2022. SwinPest: Swin Transformer-based pest recognition for precision agriculture. *Computers and Electronics in Agriculture* 198, 107044.
- [4] Deng, L., Wang, Y., Han, Z., Yu, R., 2018. Research on insect pest image detection and recognition based on bio-inspired methods. *Biosystems Engineering* 169, 139–148.
- [5] Deutsch, C.A., Tewksbury, J.J., Tigchelaar, M., et al., 2018. Increase in crop losses to insect pests in a warming climate. *Science* 361 (6405), 916–919.
- [6] Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al., 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In: *ICLR*.
- [7] FAO, 2023. *Rice Market Monitor*, Volume XXVI, Issue No. 1. Food and Agriculture Organization of the United Nations, Rome.
- [8] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: *CVPR*. pp. 770–778.
- [9] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: *CVPR*. pp. 4700–4708.
- [10] Jung, J., Maeda, M., Chang, A., et al., 2021. The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems. *Current Opinion in Biotechnology* 70, 15–22.
- [11] Kamilaris, A., Fonts, A., Prenafeta-Boldú, F.X., 2019. The rise of blockchain technology in agriculture and food supply chains. *Trends in Food Science & Technology* 91, 640–652.
- [12] Li, W., Zheng, T., Yang, Z., et al., 2020. Classification and detection of insects from field

- images using deep learning for smart pest management: A systematic review. *Ecological Informatics* 66, 101460.
- [13] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: *ICCV*. pp. 2980–2988.
 - [14] Lin, Q., Wang, H., Pei, X., Wang, J., 2020. Food safety traceability system based on blockchain and EPCIS. *IEEE Access* 7, 20698–20707.
 - [15] Liu, J., Wang, X., 2021. Plant diseases and pests detection based on deep learning: A review. *Plant Methods* 17 (1), 1–18.
 - [16] Liu, Z., Lin, Y., Cao, Y., et al., 2021. Swin Transformer: Hierarchical vision transformer using shifted windows. In: *ICCV*. pp. 10012–10022.
 - [17] Liu, Z., Mao, H., Wu, C.Y., et al., 2022. A ConvNet for the 2020s. In: *CVPR*. pp. 11976–11986.
 - [18] Nagasubramanian, K., Jones, S., Singh, A.K., et al., 2019. Plant disease identification using explainable 3D deep learning on hyperspectral images. *Plant Methods* 15, 98.
 - [19] Nanni, L., Maguolo, G., Pancino, F., 2020. Insect pest image detection and recognition based on bio-inspired methods. *Ecological Informatics* 57, 101089.
 - [20] Oerke, E.C., 2006. Crop losses to pests. *The Journal of Agricultural Science* 144 (1), 31–43.
 - [21] Patil, A.S., Tama, B.A., Park, Y., Rhee, K.H., 2020. A framework for blockchain-based secure smart greenhouse farming. *Sensors* 20 (11), 3044.
 - [22] Pattnaik, G., Shrivastava, V.K., Parvathi, K., 2020. Transfer learning-based framework for classification of pest in paddy crops. *Indian Journal of Science and Technology* 13 (42), 4425–4434.
 - [23] Petsiuk, V., Das, A., Saenko, K., 2018. RISE: Randomized input sampling for explanation of black-box models. In: *BMVC*.
 - [24] Ren, F., Liu, W., Wu, G., 2019. Feature reuse residual networks for insect pest recognition. *IEEE Access* 7, 122544–122554.
 - [25] Ribeiro, M.T., Singh, S., Guestrin, C., 2016. Why should I trust you?: Explaining the predictions of any classifier. In: *KDD*. pp. 1135–1144.
 - [26] Salah, K., Nizamuddin, N., Jayaraman, R., Omar, M., 2019. Blockchain-based soybean traceability in agricultural supply chain. *IEEE Access* 7, 73295–73305.
 - [27] Selvaraju, R.R., Cogswell, M., Das, A., et al., 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: *ICCV*. pp. 618–626.

- [28] Singh, D., Jain, N., Jain, P., et al., 2021. PlantDoc: A dataset for visual plant disease detection. In: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. pp. 249–253.
- [29] Srivastava, N., Chopra, G., Jain, P., Khatter, B., 2014. Pest monitor and control system using wireless sensor network with special reference to RPW. In: International Conference on Signal Processing and Communication. pp. 45–51.
- [30] Tan, M., Le, Q., 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICML. PMLR, pp. 6105–6114.
- [31] Thenmozhi, K., Reddy, U.S., 2019. Crop pest classification based on deep convolutional neural network and transfer learning. Computers and Electronics in Agriculture 164, 104906.
- [32] Tian, F., 2017. A supply chain traceability system for food safety based on HACCP, blockchain and internet of things. In: International Conference on Service Systems and Service Management. IEEE, pp. 1–6.
- [33] Torky, M., Hassanein, A.E., 2020. Integrating blockchain and the Internet of Things in precision agriculture: Analysis, opportunities, and challenges. Computers and Electronics in Agriculture 178, 105476.
- [34] Wang, R., Liu, L., Xie, C., et al., 2021. AgriPest: A large-scale domain-specific benchmark for practical agricultural pest detection in the wild. Sensors 21 (5), 1601.
- [35] Wen, Y., Zhang, K., Li, Z., Qiao, Y., 2016. A discriminative feature learning approach for deep face recognition. In: ECCV. Springer, pp. 499–515.
- [36] Wu, X., Zhan, C., Lai, Y.K., Cheng, M.M., Yang, J., 2019. IP102: A large-scale benchmark dataset for insect pest recognition. In: CVPR. pp. 8787–8796.
- [37] Yun, S., Han, D., Oh, S.J., et al., 2019. CutMix: Regularization strategy to train strong classifiers with localizable features. In: ICCV. pp. 6023–6032.