JCTA Journal of Computing Theories and Applications ISSN:3024-9104

Research Article

Lightweight Hybrid Deep Learning Model for Efficient Edge-Cloud Computing: A Case Study on Poultry Disease Detection

Yudi Susanto [1], Ema Utami [1,*], Afrig Aminuddin [1], and Suwanto Raharjo [2]

[1] Department of Informatics, Faculty of Computer Science, Universitas AMIKOM Yogyakarta, Sleman 55283, Indonesia; e-mail : y.susanto262@students.amikom.ac.id; ema.u@amikom.ac.id; afrig@amikom.ac.id
[2] Department of Informatics, Faculty of Science and Information Technology, Universitas AKPRIND Indonesia, Yogyakarta 55222, Indonesia; e-mail : wa2n@akprind.ac.id
* Corresponding Author : Ema Utami

Abstract: Poultry disease classification from dropping images remains challenging in resource-constrained farming environments, where visually similar symptoms complicate diagnosis and limited connectivity makes raw-image transmission costly. Although deep learning can automate classification, achieving a practical balance between model compactness, diagnostic performance, and communication efficiency remains a challenge. This study proposes a hybrid edge–cloud framework that combines a modified SqueezeNet backbone with Global Concat Pooling (GCP) for feature extraction and a Trio-Boosting classifier comprising XGBoost, LightGBM, and CatBoost with Weighted Soft Voting. Feature extraction is performed locally, while compact feature vectors are transmitted to a cloud endpoint for classification. Experiments show that GCP provides the principal performance improvement in the staged ablation, substantially outperforming conventional Global Average Pooling, whereas Trio-Boosting offers only a marginal gain over CatBoost alone. Comparisons with simpler classifiers likewise reveal no statistically significant advantage for the ensemble. In a separately trained final-model run, the framework achieved 89.23% accuracy and a weighted F1-score of 0.8924 on the test set; however, this result was not reproduced consistently during retraining and should therefore be interpreted cautiously. The backbone contains approximately 0.39 million parameters and demonstrates performance comparable to MobileNetV2 under the evaluated protocol, although it does not match the accuracy of fine-tuned ResNet50. The split-inference design reduced the transmitted payload by approximately 75%, while measured end-to-end latency remained substantially higher than model-side computational latency because of network communication. These findings suggest that the framework's primary value lies in compact feature extraction and reduced transmission payload rather than superior classification accuracy. Duplicate images and conflicting labels in the dataset, together with the use of a laptop-class CPU to emulate edge processing, limit the strength of generalization and deployment claims. Further evaluation on independent datasets and resource-constrained hardware under field conditions is required.

Keywords: Edge-Cloud Computing; Ensemble Learning; Global Concat Pooling; Hybrid Deep Learning; Poultry Disease Detection; Smart Farming; Split Inference; SqueezeNet.

Received: August, 24th 2026
Revised: October, 16th 2026
Accepted: October, 20th 2026
Published: October, 25th 2026

Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) licenses (https://creativecommons.org/licenses/by/4.0/)

1. Introduction

The poultry industry plays an important role in food security [1], yet diseases such as Coccidiosis, Newcastle disease (NCD), and Salmonella continue to threaten livestock productivity and impose substantial economic costs [2]–[4]. Delayed detection and the limitations of manual diagnosis, including its time-consuming and subjective nature, further complicate disease management, particularly when different diseases exhibit visually similar characteristics. Computer vision and deep learning have consequently attracted increasing attention as tools for automated disease classification and agricultural monitoring [5]–[7]. However, translating

these advances into practical applications for small-scale poultry farming remains challenging because classification performance must be balanced against computational requirements and communication constraints. High-capacity models such as ResNet50 can deliver strong classification performance but require substantial storage and computing resources [1], whereas lightweight architectures are more suitable for resource-constrained environments but may not preserve sufficient spatial texture information to distinguish visually similar diseases [2]. This trade-off becomes more consequential in rural areas with unreliable internet connectivity, where cloud-dependent systems that transmit raw images can incur substantial communication overhead and become less practical to operate [3]. Accordingly, an effective poultry disease classification framework must consider not only predictive performance but also the compactness of its feature representation and the amount of data transferred during inference. Power consumption, latency, memory requirements, and data privacy are also relevant considerations when deploying deep learning on edge devices [8]–[10].

To address these challenges, this study investigates a hybrid edge–cloud framework that combines a modified SqueezeNet backbone, Global Concat Pooling (GCP), and a Trio-Boosting classifier. SqueezeNet is selected as a compact feature extractor that retains spatial feature maps for subsequent pooling [11]. GCP combines Global Average Pooling (GAP) and Global Max Pooling (GMP) to represent complementary characteristics of feature activations: their average distribution and their strongest responses. This design is motivated by previous findings on CNN architecture modifications and feature-fusion approaches[12]–[14], while its effectiveness for distinguishing visually similar poultry diseases requires empirical assessment. At the classification stage, Trio-Boosting integrates XGBoost, LightGBM, and CatBoost through Weighted Soft Voting. Although combining classifiers with different tree-building strategies may offer complementary decision behavior, the resulting ensemble does not necessarily outperform an individual classifier, particularly when training data are limited. Therefore, this study evaluates Trio-Boosting against the Softmax head and simpler classifiers rather than assuming that ensemble learning provides an inherent performance advantage. In addition, the framework adopts split inference, in which feature extraction is performed locally, and only the resulting feature vector is transmitted to the cloud for classification. This design provides a basis for examining the relationship between feature compactness, communication payload, and inference latency. The edge-side evaluation, however, uses a laptop-class CPU to emulate the edge role and does not constitute validation on dedicated resource-constrained hardware.

The main contributions of this study are as follows:

- Global Concat Pooling evaluation: GCP is empirically evaluated against conventional pooling strategies through a staged ablation study to determine its contribution to classification performance.
- Comparative classifier evaluation: Trio-Boosting is compared with its individual boosting classifiers and simpler classifiers using the extracted feature representations to assess whether ensemble learning provides measurable benefits in this setting.
- Edge–cloud split-inference evaluation: The proposed framework is evaluated in terms of transmitted payload and computational latency, distinguishing model-side processing time from end-to-end latency to examine the respective contributions of computation and network communication.

Together, these contributions provide an empirical assessment of the trade-offs among feature representation, classifier selection, and transmission efficiency, while recognizing the limitations associated with dataset quality, result reproducibility, and the absence of evaluation on dedicated resource-constrained edge hardware.

The remainder of this paper is organized as follows. Section 2 reviews related research on lightweight architectures, edge–cloud deployment, and hybrid classification approaches, establishing the context for the design choices investigated in this study. Section 3 describes the proposed framework and experimental procedures, including data preparation, feature extraction, classifier training, and evaluation protocols. Section 4 presents and discusses the experimental results, focusing on the effects of pooling and classifier selection, comparisons with baseline models, and the efficiency of split inference. Finally, Section 5 summarizes the principal findings, discusses the study's limitations, and outlines directions for further research.

2. Related Work

Poultry health management has seen rapid progress in the last decade with the use of computer vision, moving from visual observation to detection systems based on deep learning. Although the identification of diseases through images of poultry droppings has shown the effectiveness of standard CNN architectures in several previous studies, their implementation on microcontrollers with limited resources remains the main challenge, as these models still require substantial computing resources. Therefore, research exploring lightweight architectures and disease detection strategies that yield precise diagnoses with operational efficiency in small-scale poultry farming environments is needed.

2.1. Heavy vs. Lightweight Architecture Trade-offs

Many previous studies used deep learning architectures with a large number of parameters to achieve maximum diagnostic accuracy: the integration of YOLOv5 and ResNet50 achieved 99% accuracy in poultry health diagnosis [15]; ResNet50 trained on 10,500 chicken dropping images yielded 98.7% accuracy [1]; and similar high-accuracy results have been reported for VGG16 and Xception through fine-tuning [5], as well as for heavyweight detectors such as YOLOv11n on class-imbalanced datasets [3]. This high accuracy stems from sophisticated feature extraction capable of mapping nonlinear patterns in biological data [2], with related success also shown in other agricultural domains such as plant leaf spot detection [16], [17]. However, these architectures require high computing resources, making them difficult to implement in environments with limited connectivity or low-end mobile devices commonly used by small-scale farmers [1], [3], and they face challenges extracting spatial features without losing micro-details once compressed for edge-oriented deployment.

Lightweight alternatives such as MobileNetV2 (26 MB) offer more stable generalization than VGG16 and Xception at lower computational cost and are recommended for regions with limited internet infrastructure [3]. Nonetheless, visually similar disease textures can still reduce classification accuracy because of feature overlap [4], [5]. In leukemia telediagnosis, MobileNetV2 was adopted as a lightweight alternative to ConvNet, ResNet50, and VGG19 [18]. The system uploads full microscopic images to a cloud-hosted web application, without reporting any reduction in transmission payload. In contrast, the present study transmits only a compact feature vector. Pooling layer optimization has been investigated as a means of addressing feature overlap, with adaptive pooling methods reported to capture more detailed feature information than standard pooling [19], [20]. Global Average Pooling (GAP) has also been shown to reduce parameter count and the risk of overfitting [13], [21]. These findings highlight the need to balance classification accuracy with computational efficiency and communication requirements when designing models for resource-constrained deployment[1], [3].

Lightweight alternatives have also been explored in the same repository, for example, an artificial neural network on handcrafted multi-color-space features that is reported to reach 97.04% accuracy without GPU support, although it combined the farm and laboratory labeled subsets (8,067 images) and used a different evaluation protocol [22]. A later study noted that the models of [5] had not been evaluated on data from a different context and reported that the accuracy of multiclass classifiers dropped to about 0.69–0.70 on unseen data from Malawi [4]; the present study shares this limitation.

2.2. IoT Integration and Infrastructure Challenges

The integration of the Internet of Things (IoT) into poultry farming enables continuous, AI-based health monitoring as an alternative to manual visual observation [2], [7]. These systems can support the early detection of environmental changes and abnormal animal behavior, with relevant information communicated directly to farmers [3], while mobile applications can reduce dependence on remotely located experts [1]. However, limited internet connectivity remains a major obstacle to practical deployment. Cloud-dependent systems may experience delayed responses or interruptions in disease detection [3], while connectivity constraints in small-scale farming environments can increase operational costs [1]. Given the rapid spread of infectious poultry diseases, timely processing is particularly important, making communication latency a critical consideration in these settings [2].

One potential strategy for addressing these constraints is to transmit compact feature representations rather than high-resolution raw images, thereby reducing the volume of data

transferred over limited-bandwidth networks [9], [23], [24]. Following this approach, the proposed edge–cloud architecture performs feature extraction locally and transmits only the resulting compact feature vectors to a cloud server for classification. This design is consistent with previous findings on split-learning approaches that can reduce communication overhead and latency in constrained wireless networks [23]. It also reflects the broader objective of coordinating AI processing with IoT infrastructure to accommodate the resource limitations of livestock farming environments [3], [15].

2.3. The Hybridization Approach as a Classification Solution

High-accuracy object detection and classification algorithms, such as YOLO and ResNet50 [1], can still produce prediction errors when conventional CNNs encounter visually similar features in healthy and early-symptomatic images [25]. This limitation motivates the integration of deep learning and machine learning techniques as an alternative classification strategy [2]. In this context, Trio-Boosting combines the class probabilities of three gradient boosting decision tree classifiers operating on features extracted by the SqueezeNet backbone, potentially exploiting their complementary decision patterns. Previous studies have reported that combining feature representations with ensemble classifiers can improve performance in automated diagnostic systems [26]. In [27], an ensemble of InceptionV3, ResNet50, and VGG16, combined with oversampling and data augmentation, was reported to outperform individual networks in multiclass skin lesion classification. Unlike this image-level ensemble of deep learning backbones, the present study applies Weighted Soft Voting to XGBoost, LightGBM, and CatBoost using a compact 512-dimensional feature vector extracted by SqueezeNet. The effectiveness of this approach is evaluated against the Softmax head and simpler classifiers rather than assuming that ensemble learning necessarily improves classification performance.

Building on the broader feature-fusion and ensemble-learning approach discussed in [26], the proposed framework applies Weighted Soft Voting to XGBoost, LightGBM, and CatBoost using features extracted by the SqueezeNet backbone. The ensemble is evaluated against the Softmax head and simpler classifiers to determine whether combining these classifiers improves the discrimination of visually similar poultry disease features, while retaining lightweight feature extraction on the edge device. Existing studies reveal two related challenges. High-accuracy architectures such as ResNet50 and YOLOv5 have reported accuracy above 98% [15], but their computational and storage requirements can constrain deployment on resource-limited devices and increase reliance on cloud infrastructure, particularly where network connectivity is unstable[2], [3]. Conversely, lightweight architectures such as MobileNetV2 reduce computational requirements but may still struggle to distinguish visually similar textures, potentially limiting classification performance[5]. These observations highlight the need to evaluate lightweight feature extraction, classifier effectiveness, and communication efficiency jointly rather than treating classification accuracy as the sole indicator of practical suitability.

Table 1. Comparison of Related Studies

Author	Methodology	Limitation
Machuve et al. [5]	MobileNetV2 & Xception	Large transmitted payload (raw image transmission)
Degu and Simegn [1]	YOLO-v3 & ResNet50	Resource-heavy (~90MB); GPU dependent
Ali et al. [15]	YOLO-v5 & ResNet50	Transmits full (scrambled) images to a server; latency and connectivity not evaluated
Dhungana [3]	YOLO-v11 + EfficientNet	Requires a high-end GPU (RTX 4000)
Natsir et al. [28]	AI Systematic Review	Lack of large-scale validation studies
Tusar et al. [18]	MobileNetV2 (cloud web telediagnosis)	Full-image upload; payload reduction and edge-side evaluation not reported
Zaw and Mon [27]	Inception V3 + ResNet 50 + VGG16 ensemble	Heavy image-level backbones; not designed for edge deployment
Proposed	Hybrid SqueezeNet	Small-scale, single-source dataset; requires validation on physical edge hardware

As summarized in Table 1, previous studies have explored diverse architectures and classification strategies, but the reported limitations indicate that model accuracy alone is insufficient to establish deployment suitability in rural poultry farming environments. In particular, the extent to which a lightweight backbone can preserve discriminative features, whether ensemble classification provides measurable improvements over simpler alternatives, and how compact feature transmission affects end-to-end latency require further empirical investigation. To address these issues, this study evaluates Global Concat Pooling (GCP) within the SqueezeNet backbone and adopts a split-inference edge–cloud architecture that transmits extracted feature vectors rather than raw images. A staged ablation study (Section 3.5) further examines the contributions of the architectural and classification choices, providing a more systematic assessment than evaluating the combined framework solely as a single configuration.

3. Proposed Method

To optimize the use of computing resources and improve disease detection accuracy, this study developed the poultry disease detection model using the SqueezeNet architecture with a Global Concat Pooling Layer as the feature extractor, combined with Trio-Boosting consisting of XGBoost, LightGBM, and CatBoost as the disease classifiers.

Fig. 1 maps the technical pipeline in five stages: (1) preprocessing and balancing, in which the stratified split precedes oversampling and augmentation, applied to the training partition only; (2) lightweight feature extraction on the edge side, where the modified SqueezeNet backbone with Global Concat Pooling (GCP) produces a 512-dimensional feature vector; (3) split-inference transmission of this float32 feature vector from the edge node to the cloud endpoint; (4) Trio-Boosting classification on the cloud server, where the class probabilities of XGBoost, LightGBM, and CatBoost are combined by Weighted Soft Voting; and (5) evaluation by classification metrics, staged ablation, model-side and end-to-end latency measurements, and TFLite quantization tests.

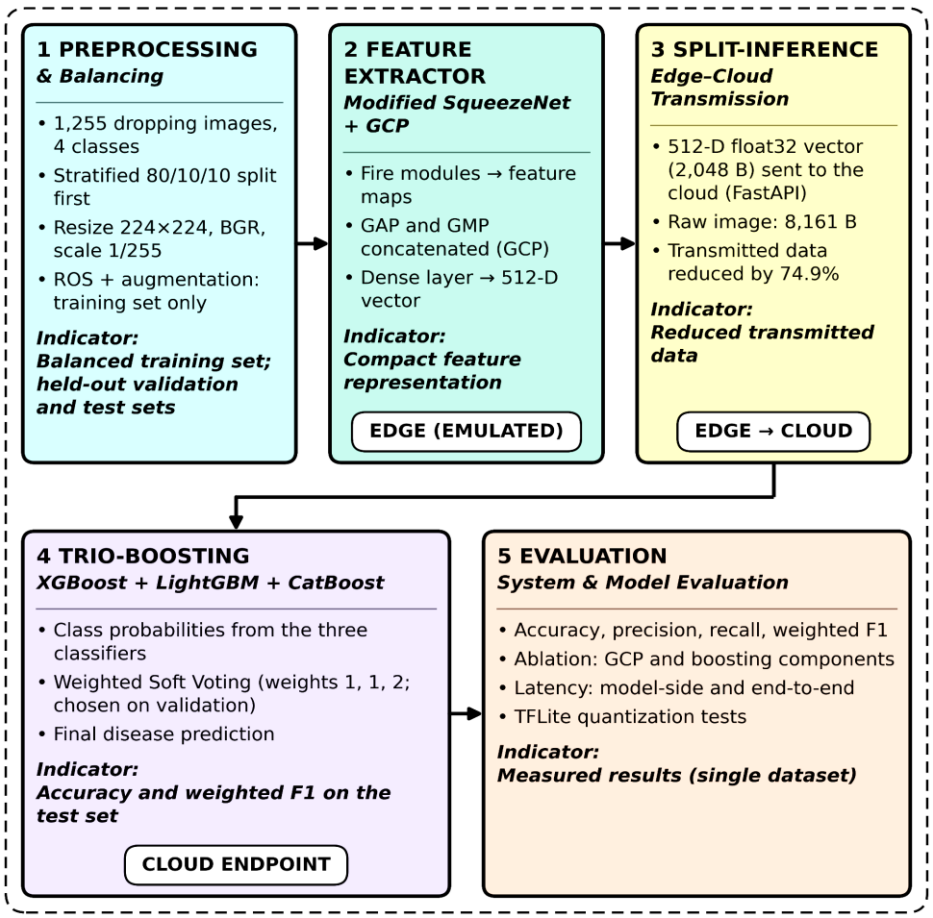


Figure 1. Research Workflow

Table 2 summarizes, for each component, the limitation it addresses, the design choice, and the supporting evidence.

Table 2. Design rationale and corresponding evidence for each component

Component	Limitation addressed	Design choice	Evidence
SqueezeNet	Heavy CNNs exceed edge memory and computation	Fire modules (1×1 squeeze, 1×1/3×3 expand)	Parameters and size vs. baselines (Tables 7 and 11)
Global Concat Pooling	GAP dilutes localized textures; GMP discards their spread	Concatenate GAP and GMP (256 + 256 → 512)	GAP vs. GMP vs. GCP (Table 4)
XGBoost, LightGBM, CatBoost	Softmax head may limit discrimination of similar textures; different tree strategies may make different errors	Three GBDT classifiers on the 512-dim features	Single classifiers vs. Softmax (Table 5) and vs. simple classifiers (Table 6); gain modest
Weighted soft voting	A single classifier may be unreliable on small data	Probability averaging with validation-selected weights (final model: 1, 1, 2)	Ensemble vs. best single classifier (Table 5) and vs. simple classifiers (Table 6); no significant gain
Edge–cloud split inference	Raw-image upload is heavy under limited connectivity	Backbone and GCP on the edge side; only the 512-dimensional float32 feature vector is sent to the cloud classifier	Transmitted data per inference; model-side vs. end-to-end latency (Tables 12-13)

3.1. Dataset Characteristics

This study uses the laboratory-labeled subset of a publicly accessible, annotated Zenodo dataset of poultry dropping images, comprising 1,255 images across four distinct classes: Coccidiosis (373 images), Healthy (347 images), Newcastle Disease (NCD, 186 images), and Salmonella (349 images) [29]. To rigorously evaluate model performance and prevent data leakage, a strict experimental sequence was implemented. First, a stratified 80/10/10 split was applied independently within each class to form the train, validation, and test sets. This yields the following per-class distributions across partitions: Coccidiosis (298 train, 37 validation, 38 test; total 373), Healthy (277 train, 34 validation, 36 test; total 347), Newcastle Disease / NCD (148 train, 18 validation, 20 test; total 186), and Salmonella (279 train, 34 validation, 36 test; total 349), summing to an overall distribution of 1,002 training, 123 validation, and 130 test images. This splitting step strictly preceded any data balancing (Random Oversampling) or data augmentation, so that the validation and test partitions contain no oversampled or augmented copies of training images. The source dataset contains a small number of near-identical images (e.g., repeated shots of the same sample). A post hoc check identified 35 duplicate pairs, of which 19 pairs contained identical images assigned different class labels: 16 Coccidiosis–Healthy pairs and three Healthy–Salmonella pairs. The impact of this residual overlap and the associated label inconsistencies is examined in Section 4.8.1.

Per the Zenodo documentation, all image labels were annotated by the NM-AIST research team using Polymerase Chain Reaction (PCR)-based molecular laboratory testing rather than visual judgment alone, which is intended to provide biologically validated ground-truth labels.

3.2. Image Preprocessing and Normalization

Before training, all images were resized to 224×224 pixels using bilinear interpolation to match the input dimensions used by the SqueezeNet architecture (Equation 1). The resized images were then converted to BGR format and normalized to the range [0, 1] using 1/255 as defined in Equation 2. The original images had been downsampled to 96×96 pixels during dataset preparation and were subsequently upsampled to 224×224 pixels. This upsampling does not restore fine-grained information lost during the initial downsampling; its potential implications for feature extraction are discussed in the Limitations section. The coordinate mapping for resizing, with bilinear interpolation applied at non-integer source coordinates, is expressed as follows:

$$I_{resized}(x', y') = I_{in}\left(x' \cdot \frac{W_{in}}{224}, y' \cdot \frac{H_{in}}{224}\right) \quad (1)$$

$$I_{norm}(x', y') = \frac{I_{resized}(x, y)}{255} \quad (2)$$

where W_{in} , H_{in} denote the input image width and height, respectively, and I_{norm} represents the normalized image supplied to SqueezeNet.

Random oversampling (ROS) was applied exclusively to the training partition after the dataset had been split, as illustrated in Fig. 2. Consequently, oversampling did not introduce additional copies of training images into the validation or test partitions. Exact duplicate images present in the original dataset originated from the source repository rather than from the oversampling procedure. For each class $i \in 1,2,3,4$, the target sample count N_{target} was set to the size of the largest class in the training partition, namely 298 images for Coccidiosis. Minority classes were balanced by sampling with replacement from their respective training images. Let N_i denote the number of training images in class i, and let S_i denote the corresponding set of images. The additional samples R_i were drawn uniformly from S_i to obtain $N_{target} - N_i$ additional instances per minority class. This procedure produced a balanced training set of 1,192 images, comprising 298 images per class. Partitioning the dataset before oversampling helps prevent leakage caused by the resampling procedure and is consistent with established experimental protocols that restrict resampling to the training data [18], [27]. The target sample count and the resulting resampled training set are defined as[30].

$$N_{target} = \max(N_1, N_2, N_3, N_4) \quad (3)$$

$$S'_i = S_i \cup R_i, \qquad |R_i| = N_{\text{target}} - N_i \quad (4)$$

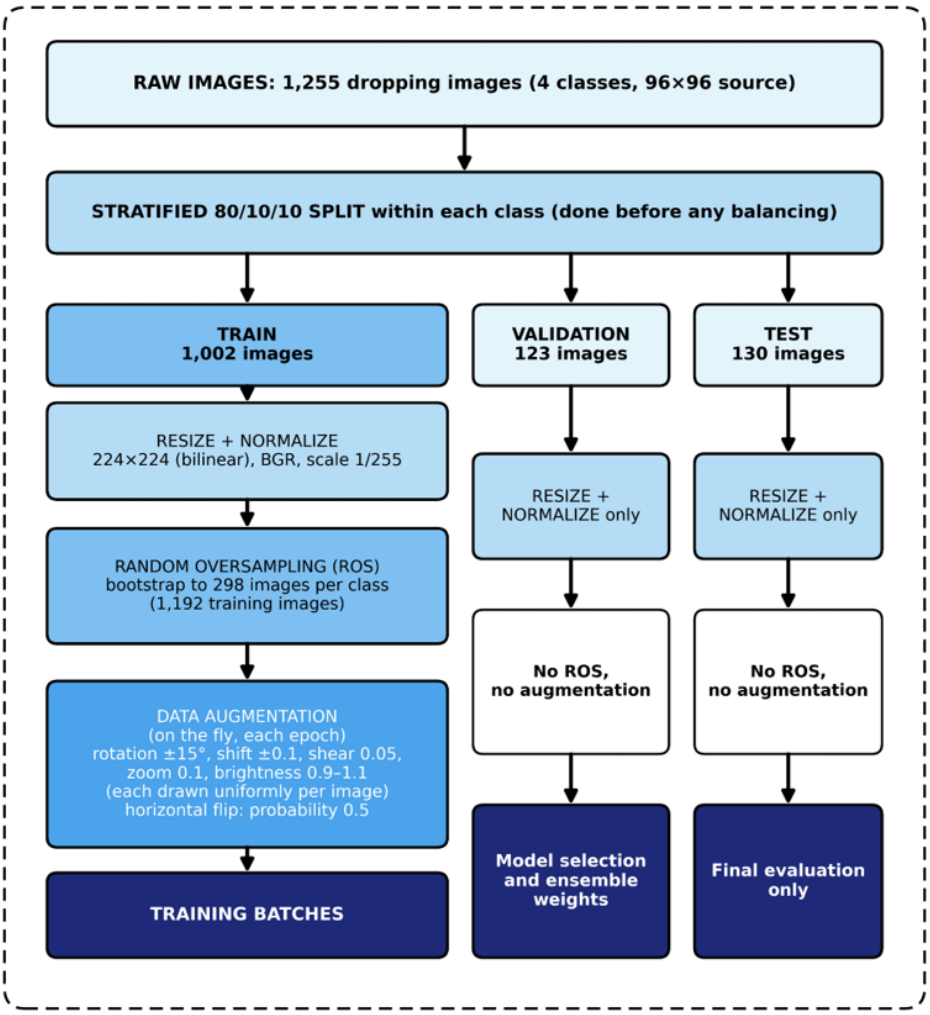


Figure 2. Data preprocessing and partitioning sequence

3.3. Data Augmentation

To improve the model's ability to generalize to varying field conditions, geometric and brightness transformations were applied exclusively to the training images after oversampling. The validation and test sets were not augmented, allowing model performance to be evaluated on images without artificially generated transformations. Augmentation was performed on the fly during training, with pixels outside the original image boundaries filled using the nearest pixel values.

The following transformations were applied to each training image using randomly sampled parameter values: rotation by an angle uniformly distributed between $-15°$ and $15°$, horizontal and vertical shifts uniformly distributed between -0.1 and 0.1 of the corresponding image dimensions, shear within a range of $\pm 0.05°$, zoom factors between 0.9 and 1.1, and brightness scaling factors between 0.9 and 1.1. Horizontal flipping was applied with a probability of 0.5. These transformations were implemented using Keras ImageDataGenerator. The training script used to produce the final model was not preserved; therefore, the random seeds for augmentation and weight initialization could not be recovered. The gradient boosting decision tree (GBDT) classifiers in the final model used seed 42, whereas repeated baseline experiments used seeds 42, 43, and 44 for oversampling, weight initialization, batch ordering, and augmentation.

The augmentation strategy was intended to expose the model to variations in camera angle, image position, scale, and lighting that may occur in poultry farming environments, while retaining the image format required by SqueezeNet [11]. However, the model's robustness to these variations was not evaluated independently. Previous research has highlighted the importance of acquisition conditions, feature representation, and realistic evaluation protocols in assessing model robustness, including evaluation across different image acquisition conditions[31].

3.4. SqueezeNet Architecture with Global Concat Pooling

SqueezeNet was selected as the feature extractor because its compact architecture is suitable for resource-constrained environments where memory and computational capacity are limited. Its Fire modules reduce the number of parameters by using a squeeze layer with 1×1 convolutions followed by parallel 1×1 and 3×3 expand convolutions. This design substantially reduces model size compared with AlexNet while retaining spatial feature maps for subsequent feature extraction [11].

The implemented backbone is a truncated SqueezeNet v1.1-style network trained from scratch with randomly initialized weights. Input images were resized to 224×224×3 pixels using bilinear interpolation. The network begins with a 3×3 convolutional layer containing 64 filters and a stride of 2, followed by max pooling and batch normalization. Four sequential Fire modules (Fire2–Fire5) are then applied, with squeeze/expand filter configurations of 16/64, 16/64, 32/128, and 32/128, respectively. Additional max pooling is applied after Fire3, and batch normalization is applied after each squeeze layer. The original Conv10 classification layer is removed, allowing Fire5 to produce a spatial feature map of size 28×28×256. Each Fire module first compresses its input feature maps through a squeeze convolution, as defined in Equation (5). The resulting representation is processed by parallel expand convolutions, and their outputs are concatenated along the channel dimension, as expressed in Equation (6):

$$S = \sigma(W_s * x + b_s) \tag{5}$$

$$E = [\sigma(W_{e1} * S + b_{e1}),\ \sigma(W_{e3} * S + b_{e3})] \tag{6}$$

where W_s, W_{e1}, W_{e3} denote convolutional filters; b_s, b_{e1}, b_{e3} are the corresponding biases; σ denotes the ReLU activation function; $*$ denotes convolution; and the brackets represent channel-wise concatenation.

To obtain a compact yet potentially discriminative feature representation, Global Concat Pooling (GCP) combines Global Average Pooling (GAP) and Global Max Pooling (GMP). GAP captures the average activation of each feature channel but may dilute localized responses, whereas GMP preserves the strongest activation but does not represent how broadly that response occurs. Combining the two representations is intended to retain complementary information about the overall distribution and peak intensity of feature activations without

introducing additional trainable parameters in the pooling operation. This design is motivated by previous work on pooling strategies and feature fusion [13], [14], [32], [33].

For the final feature map F, with spatial dimensions $M \times N$ and channel index c, GAP and GMP are calculated as follows:

$$V_{\text{avg},c} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} F_c(i,j), \qquad c = 1,2,\ldots,256 \tag{7}$$

$$V_{\max,c} = \max_{i,j} F_c(i,j), \qquad c = 1,2,\ldots,256 \tag{8}$$

The two vectors are concatenated to produce the GCP representation:

$$V_c = [V_{avg}, V_{max}] \tag{9}$$

Because each pooled vector contains 256 elements, their concatenation produces a 512-dimensional representation regardless of the spatial resolution of the input feature map. This compact representation is suitable for the proposed split-inference architecture, in which feature extraction takes place on the edge device and the resulting feature vector is transmitted to the cloud for classification.

The concatenated vector is subsequently passed through a fully connected layer with ReLU activation to obtain the final 512-dimensional deep feature vector:

$$V_{deep} = \sigma(W_d \cdot V_c + b_d) \tag{10}$$

where W_d and b_d denote the weights and bias of the fully connected layer, respectively. The resulting feature vector is supplied to the boosting classifiers used in Trio-Boosting. During training of the Softmax classification head, batch normalization and dropout with a rate of 0.6 are applied after the dense layer. The reported backbone contains 387,812 parameters, comprising 386,468 trainable parameters and 1,344 non-trainable batch-normalization parameters.

GCP was investigated as a relatively simple alternative to more complex attention mechanisms, including dual attention [34] and channel attention [25]. Its effectiveness in distinguishing visually similar poultry disease classes was evaluated through the pooling ablation experiments reported in Table 4. The contribution of GCP is therefore treated as an empirical question rather than an assumed improvement.

3.5. Experimental Design for Ablation Study

A staged, seven-stage experimental design was used to evaluate the effects of successive architectural and classification configurations. The first six stages examined the pooling strategy and classifier selection, while Stage 7 evaluated post-training quantization for edge-oriented deployment. Stage 1 used SqueezeNet with Global Average Pooling (GAP) and a Softmax classifier as the baseline; a Global Max Pooling (GMP) variant was also evaluated for comparison (Table 4). Stage 2 replaced the baseline pooling configuration with Global Concat Pooling (GCP). In Stages 3–5, the Softmax classifier was replaced separately with XGBoost, LightGBM, and CatBoost, respectively, with each configuration compared against the Stage 2 baseline. Stage 6 combined the three boosting classifiers using Weighted Soft Voting. Stage 7 evaluated float16, dynamic-range 8-bit, and full-integer INT8 quantization on the final model rather than within the same ablation run. Accordingly, the quantization results are treated as an additional deployment-oriented evaluation, not as a directly comparable stage in the architectural ablation.

All six primary ablation stages followed the same data preparation and evaluation protocol. Random Oversampling (ROS), as defined in Equations (3)–(4), and data augmentation were applied to the training partition only. Early stopping with a minimum of 15 epochs was used to support generalization, and ensemble weights were selected through grid search on the validation set. The test set was reserved for final evaluation to avoid using test results for model or hyperparameter selection. Although this staged design facilitates comparison between successive configurations, it does not independently isolate every component's contribution, as a full leave-one-component-out ablation would.

For reproducibility, a fixed random seed of 42 was used for the final GBDT classifiers and the Random Forest baseline. The principal hyperparameters and their corresponding

rationales are summarized in Table 3. All three boosting classifiers used multiclass objectives: multi:softprob for XGBoost, multiclass for LightGBM, and multiclass log loss for CatBoost. XGBoost used L2 regularization (reg_lambda = 1.0), while LightGBM used L1 and L2 regularization (reg_alpha = 0.1, reg_lambda = 0.1) together with gradient-based one-side sampling (GOSS). CatBoost used symmetric trees with L2 leaf regularization (l2_leaf_reg = 3.0). The ensemble weights were selected using validation accuracy, with the search procedure described in Section 3.6.

Table 3. Hyperparameter Configuration

Algorithm	Hyperparameter	Value	Justification
XGBoost	Learning Rate	0.05	Regularized boosting for overfitting prevention
	Max Depth	5	Capturing non-linear relationships
	N-Estimators	500	Sufficient iterations for model validation
LightGBM	Num Leaves	31	Leaf-wise growth for memory/speed efficiency
	Learning Rate	0.05	Synchronization of learning rate with XGBoost
	N-Estimators	500	Sufficient iterations for model validation
CatBoost	Depth	5	Capacity to model complex textural correlations
	Iterations	500	Learning of deep correlation features

3.6. Trio-Boosting Mechanism

The proposed Trio-Boosting classifier combines three Gradient Boosting Decision Tree (GBDT) algorithms: XGBoost, LightGBM, and CatBoost. These classifiers were selected to investigate whether their different tree-building strategies could provide complementary predictions when applied to the 512-dimensional feature vector V_{deep} extracted by the SqueezeNet–GCP feature extractor. Unlike a linear Softmax output layer, tree-based ensembles can model nonlinear interactions among feature dimensions, making them a potential alternative for classifying visually similar poultry disease patterns. A combination of these boosting algorithms has also been used for histopathology image classification[35].

The three classifiers employ different learning mechanisms. XGBoost uses a regularized objective that penalizes tree complexity to help control overfitting [36]. LightGBM employs histogram-based, leaf-wise tree growth, gradient-based one-side sampling, and exclusive feature bundling to improve computational efficiency, although leaf-wise growth may increase overfitting risk on small datasets if tree complexity is not controlled [37]. CatBoost uses ordered boosting to address prediction shift and symmetric trees to support efficient prediction [38]. These differences motivated the evaluation of a weighted ensemble, although complementary behavior among the classifiers was not assumed and its effect on individual disease classes was not assessed separately. Performance comparisons with the individual classifiers are reported in Tables 5 and 6.

For each disease class C, the predicted probabilities from XGBoost, LightGBM, and CatBoost are denoted by $P_k(C|V_{deep})$, where $k \in 1,2,3$. Weighted Soft Voting combines these probabilities rather than selecting the final class through a majority vote. This allows each classifier's contribution to depend on its assigned weight, with the objective of improving the combined prediction relative to individual classifiers [39]. The final class P_{final} probability is calculated as:

$$P_{final}(C) = \sum_{k=1}^{3} W_k \cdot P_k(C \mid V_{deep}) \tag{11}$$

where W_k denotes the normalized weight assigned to classifier k subject to $\sum_{k=1}^{3} W_k = 1$.

The ensemble weights were selected through an exhaustive grid search over integer-valued weight combinations from 0 to 5, using a step size of 1 and excluding the all-zero combination. This yielded 215 candidate combinations, which were normalized to sum to 1 before evaluation on the validation set. The combination achieving the highest validation accuracy was selected for the final model. The resulting weights were $(W_{XGB}, W_{LGBM}, W_{Cat})$ = (0.25, 0.25, 0.25). Although CatBoost received the largest weight, its individual validation accuracy was the lowest among the three classifiers at 86.99% (Table 6). This reflects the selection of

weights based on the validation accuracy of the combined predictions rather than the standalone accuracy of each classifier. The final predicted disease class is determined by selecting the class with the highest combined probability:

$$\hat{y} = \arg\max_{C} P_{\text{final}}(C) \tag{12}$$

Edge-cloud split inference: The split point is placed after the 512-dimensional feature layer: the backbone and Global Concat Pooling run on the edge side and the Trio-Boosting classifiers run in the cloud (Algorithm 1), so that only the float32 feature vector, instead of the raw image, is transmitted. This design targets limited connectivity and is evaluated through transmitted data per inference and model-side versus end-to-end latency.

Algorithm 1. Hybrid Edge-Cloud Inference for Poultry Disease Detection

REQUIRE: Raw Droppings Image | ENSURE: Predicted Disease Class

Stage 1: Edge (Local)

1: $I_{proc} \leftarrow$ Resize $(I_{raw}, 224 \times 224)$

2: F $\leftarrow$ SqueezeNet - GCP I_{proc}; $V_c \leftarrow$ [GAP(F), GMP(F)]; $V_{deep} \leftarrow Dense(V_C)$

3: Transmit V_{deep} (512-dimensional float32 feature vector) to Cloud API Stage 2: Cloud (Remote)

4: Load Weighted GBDT Ensemble Models (XGB, LGBM, Cat; weights 0.25, 0.25, 0.50)

5: $P_{final}(C) = \sum_{k=1}^{3} W_k \cdot P_k(C \mid V_{deep})$; $C_{pred} = arg\ max_c\ (P_{final})$

6: Return C_{pred} to User Interface

3.7. Model Training and Validation

As illustrated in Fig. 3, the experimental procedure consisted of two stages: training the SqueezeNet feature extractor and training the Trio-Boosting classifiers. Both stages used the same predefined data partitions.

Feature Extractor Training: SqueezeNet was trained for 100 epochs with a batch size of 16, using categorical cross-entropy loss and the Adam optimizer with an initial learning rate of 10^{-4}. ModelCheckpoint was used to retain the model with the highest validation accuracy, while ReduceLROnPlateau reduced the learning rate by half when the validation loss did not improve for seven consecutive epochs.

Trio-Boosting Training: After feature extractor training, the Softmax classification layer was removed, and the 512-dimensional feature vectors generated by the selected SqueezeNet checkpoint were used to train XGBoost, LightGBM, and CatBoost. The principal hyperparameters were selected through four-fold cross-validation, with the final configurations reported in Table 3.

Validation Procedures: Model performance was assessed using accuracy, precision, recall, and F1-score, reported at both the class level and the weighted-average level. Recall is particularly relevant to disease detection because missed cases may delay intervention and increase the risk of disease transmission. Class-wise F1-scores were also examined to identify differences in classification performance across the four disease categories.

Efficiency Measurement: Model-side computational latency was defined as the combined time for edge-side feature extraction and cloud-side classification, excluding serialization and network transmission. End-to-end latency was measured from image input to the returned prediction and included serialization and network transmission; the corresponding results are reported in Table 12. Transmission-time values in Table 13 are analytical estimates based on assumed network conditions, rather than direct measurements.

The transmitted payload per inference was defined as the data sent from the edge device to the cloud server for one prediction. The encoded 96×96 JPEG image required 8,161 bytes, whereas the 512-dimensional float32 feature vector required 2,048 bytes (512 × 4 bytes), representing a 74.9% reduction under the evaluated image format and resolution. This comparison quantifies payload size rather than network bandwidth, which is measured in bits per second.

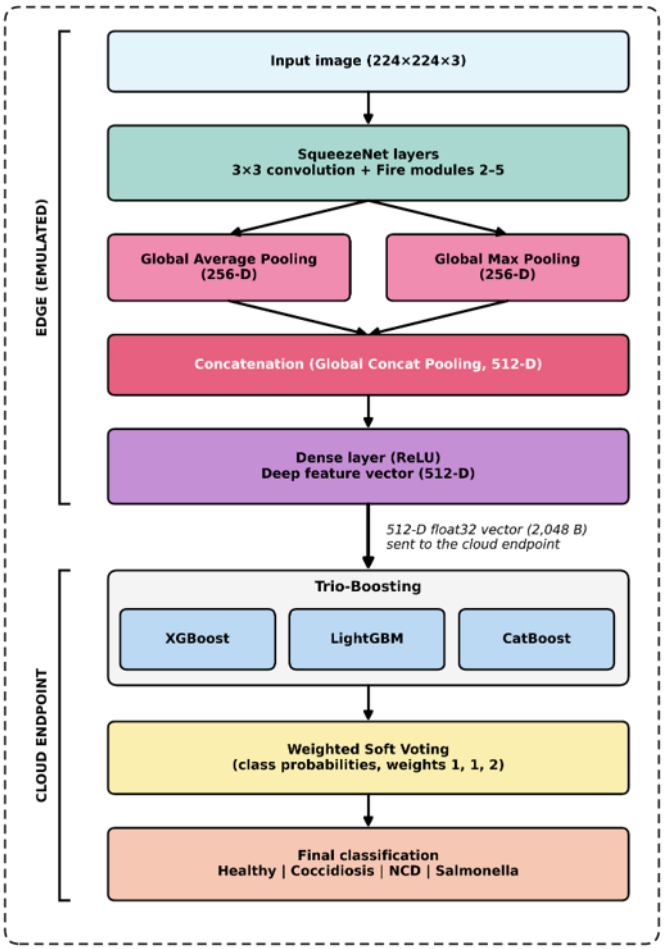


Figure 3. Architecture

4. Results and Discussion

This section presents the training and ablation results, baseline comparisons, and two deployment scenarios: full-precision model inference (Scenario 1) and TFLite-quantized feature extraction on a laptop-class CPU emulating the edge role (Scenario 2). All results are based on the 130-image test set from a single public dataset.

4.1. Training Performance and Model Stability

The training and validation curves in Fig. 4 remain close and show no clear sign of overfitting.

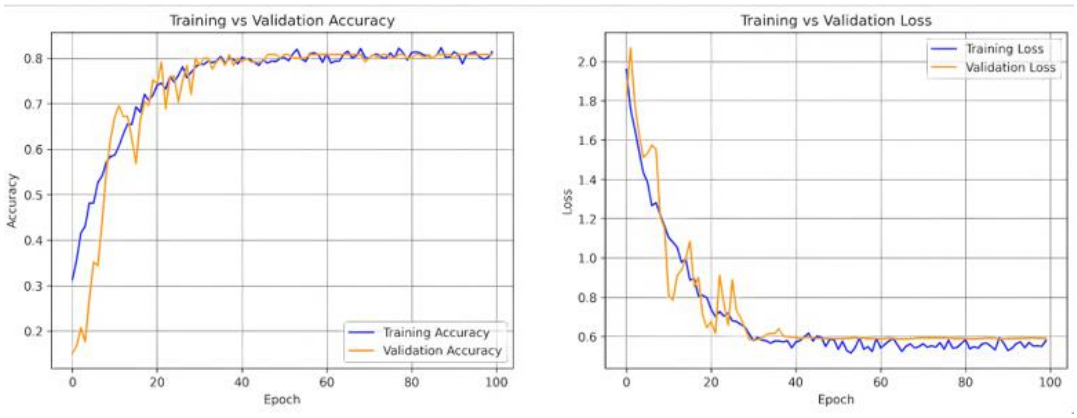


Figure 4. Accuracy and Loss curves

4.2. Global Concat Pooling Ablation Study Results

GCP was compared with single-pooling baselines under identical training conditions (Table 4). GCP reached the highest accuracy (84.62%), compared with 81.54% for GMP and 62.31% for GAP, which suggests that peak responses carry most of the disease-relevant texture, while the 3.08-point gain of GCP over GMP (about four images) is small and comes from a single run.

Table 4. Ablation Studies

Pooling Strategy	Accuracy	F1 Score (Weighted)
SqueezeNet + Global Average Pooling	62.31%	0.6260
SqueezeNet + Global Max Pooling	81.54%	0.8168
SqueezeNet + Global Concat Pooling	84.62%	0.8467

4.3. Trio-Boosting vs Single Model Classification Performance

Replacing the Softmax head with a single boosting classifier changed accuracy by −2.31 percentage points (XGBoost), −0.77 (LightGBM), and +0.76 (CatBoost), and Weighted Soft Voting equaled CatBoost at 85.38% (Table 5). The ablation gain therefore comes mainly from the GCP features (Stages 1–2) rather than from the classifier. On the final-model features, the ensemble (89.23%) did not differ significantly from the single boosting classifiers (87.69–90.00%) or the simple classifiers (84.62–87.69%) (exact McNemar test, $p \geq 0.18$; Table 6) [40]. With only 130 test images, the test has low power, so the benefit of boosting and voting over simpler classifiers was not demonstrated, and complementary decision behavior across classifiers was not evaluated separately.

Table 5. Trio-Boosting and Single Boosting Performance in the Ablation Run (simple classifiers are reported in Table 6)

Classifier Model	Accuracy	Weighted F1	Change from Stage 2 (percentage points)
Stage 2: GCP + Softmax (baseline, from Table 4)	84.62%	0.8467	-
Stage 3: + XGBoost Only	82.31%	0.8230	−2.31
Stage 4: + LightGBM Only	83.85%	0.8378	−0.77
Stage 5: + CatBoost Only	85.38%	0.8534	+0.76
Stage 6: + Weighted Soft Voting (Trio-Boosting, ablation run)	85.38%	0.8534	+0.76

Table 6. Classifier comparison on the 512-dimensional features of the final model (test n = 130; duplicate-free test n = 121; McNemar exact test against Weighted Soft Voting)

Classifier	Selected setting	Val. acc.	Test acc.	Weighted F1	Test acc. (n = 121)	McNemar p
XGBoost	depth 5	89.43%	88.46%	0.8854	88.43%	1.000
LightGBM	31 leaves	90.24%	87.69%	0.8769	88.43%	0.688
CatBoost	depth 5	86.99%	90.00%	0.9003	90.08%	1.000
Weighted Soft Voting	weights (1, 1, 2)	91.06%	89.23%	0.8924	89.26%	–
Logistic Regression	C = 0.001	87.80%	86.92%	0.8705	86.78%	0.453
SVM (RBF)	C = 100, γ = 0.0001	90.24%	84.62%	0.8475	85.12%	0.180
Random Forest	unrestricted depth	85.37%	87.69%	0.8778	88.43%	0.727

4.4. Comparison with Research Baseline

ResNet50 and MobileNetV2 were compared with the proposed model on the same split and test images in three settings: frozen ImageNet features with XGBoost; MobileNetV2

trained from scratch with a GAP head (seed 42); and end-to-end fine-tuning under the same augmentation, optimizer, schedule, and classification head as the proposed model [41], [42]. Fine-tuned models used seeds 42-44, no hyperparameter tuning, and each architecture's own ImageNet preprocessing.

Under the matched protocol, the proposed backbone with a Softmax head (83.59 ± 2.35%) was similar to fine-tuned MobileNetV2 (84.10 ± 3.79%) and below fine-tuned ResNet50 (90.77 ± 1.33%), whose three runs all exceeded the three runs of the backbone (Tables 7 and 8). The backbone has about 9 and 66 times fewer parameters than MobileNetV2 and ResNet50 (0.39 M versus 3.57 M and 25.69 M). The frozen-feature and scratch baselines were clearly lower (53.08–80.00% and 62.31%). The final model (89.23%, single run) lies above all three retrained runs and was not reproduced by retraining, so it should be read as the accuracy of one trained model rather than the expected accuracy of the architecture.

Table 7. Comparison with baselines (test n = 130)

Model	Training setting	Accuracy	Weighted F1	Parameters (M)
ResNet50	Frozen ImageNet + XGBoost	53.08%	–	25.6
MobileNetV2	Frozen ImageNet + XGBoost	80.00%	–	3.4
MobileNetV2	Scratch, GAP head, single run (seed 42)	62.31%	0.6274	2.26
MobileNetV2	Fine-tuned from ImageNet, matched protocol, 3 seeds	84.10 ± 3.79%	0.8434 ± 0.0375	3.57
ResNet50	Fine-tuned from ImageNet, matched protocol, 3 seeds	90.77 ± 1.33%	0.9067 ± 0.0138	25.69
Proposed backbone (SqueezeNet+GCP, Softmax head)	Scratch, matched protocol, 3 seeds	83.59 ± 2.35%	0.8344 ± 0.0246	0.39
Proposed (SqueezeNet+ Trio-Boosting)	Trained from scratch, single run	89.23%	0.8924	0.39

Note: the fine-tuned baselines carry the same classification head as the proposed model, which adds 1.3 M (MobileNetV2) and 2.1 M (ResNet50) parameters; the frozen MobileNetV2 count (3.4 M) includes the original ImageNet head, whereas the scratch-trained MobileNetV2 (2.26 M) is counted without it.

Table 8. Per-seed results of the fine-tuned baselines and the proposed backbone

Model	Seed	Val. acc.	Test acc.	Weighted F1	Best epoch
MobileNetV2	42	84.55%	82.31%	0.8264	37
MobileNetV2	43	90.24%	88.46%	0.8864	67
MobileNetV2	44	87.80%	81.54%	0.8174	48
ResNet50	42	91.06%	92.31%	0.9226	17
ResNet50	43	90.24%	90.00%	0.8983	7
ResNet50	44	92.68%	90.00%	0.8991	5
Proposed backbone (Softmax)	42	80.49%	83.08%	0.8294	46
Proposed backbone (Softmax)	43	86.99%	81.54%	0.8127	79
Proposed backbone (Softmax)	44	88.62%	86.15%	0.8611	66

The comparison is limited: the baselines were not tuned, the final model is a single run whereas the baselines are three-seed means, and its training script was not preserved, so the 5.6-point gap to the retrained backbone could not be explained. Because voting added only 0.76 points in the ablation, the gap is unlikely to come from the classifier alone. Parameter counts and training settings also differ across rows, as noted in Table 7.

4.5. Poultry Disease Classification Results

In Scenario 1 (cloud-only), the final model (100 epochs) achieved an accuracy of 89.23% and a weighted F1-score of 0.8924 (Table 9), with a 95% Wilson confidence interval for accuracy of 82.7–93.5% (n = 130) [43]. This run is separate from the ablation and not directly comparable to it.

Table 9. Classification Report of the Proposed Model

Class	Precision	Recall	F1-score	Support
Coccidiosis	0.9167	0.8684	0.8919	38
Healthy	0.8205	0.8889	0.8533	36
Newcastle Disease	0.9091	1.0000	0.9524	20
Salmonella	0.9394	0.8611	0.8986	36
Macro avg	0.8964	0.9046	0.8990	130
Weighted avg	0.8952	0.8923	0.8924	130

The macro-average F1-score (0.8990) is close to the weighted value (0.8924), so performance is not driven by the larger classes at the expense of NCD, which has only 20 test images. NCD was the easiest class to identify (F1-score 0.9524, all 20 test images correct). Healthy and Coccidiosis were the most confused classes: 5 Coccidiosis images were predicted as Healthy and 1 Healthy image as Coccidiosis (Fig. 5), together 6 of the 14 errors, and Salmonella accounted for 5 further errors.



Figure 5. Confusion Matrix

4.6. Hybrid Edge-Cloud Evaluation

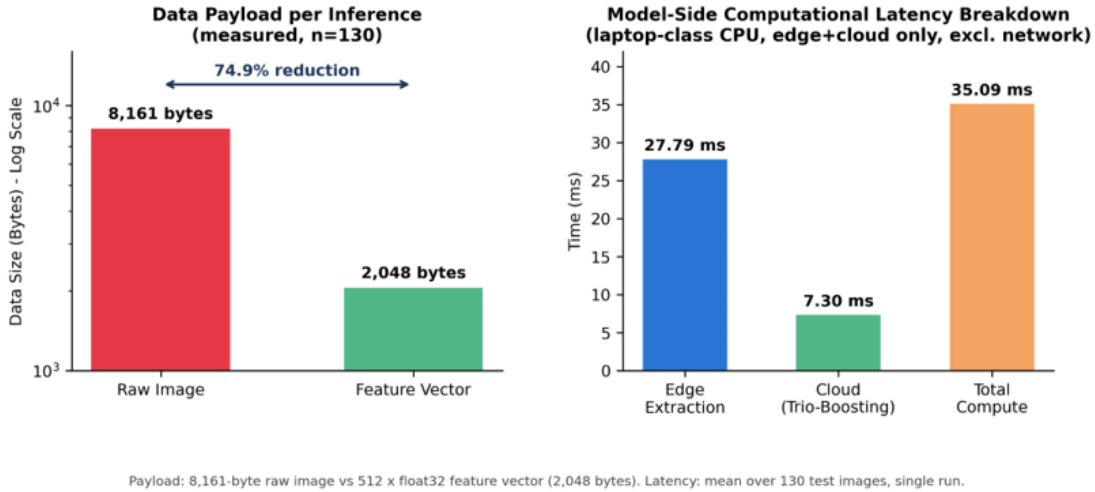


Figure 6. Transmitted data per inference and model-side computational latency (excluding network transmission)

Fig. 6 shows that sending the 512-dimensional feature vector instead of the raw image reduced the transmitted data per inference by 74.9% (8.0 KB to 2.0 KB), and that model-side computational latency was approximately 35.09 ms on a laptop-class CPU emulating the edge role. In Scenario 2, the feature extractor was converted to TFLite as float16, dynamic-range (8-bit weights), and full-integer INT8 (calibrated with 100 or 300 training images), and feature extraction ran on a laptop-class CPU (Intel x86_64 MacBook) emulating the edge role. Float16 left accuracy unchanged (89.23%), dynamic-range quantization lowered it by one image (88.46%), and full-integer INT8 by twelve images (80.00%) for both calibration sizes (Table 10). The dynamic-range model (0.42 MB) is retained for the edge role because of its smaller size, although float16 (0.783 MB) was slightly more accurate and faster in this measurement, and full-integer conversion is not recommended without further calibration.

Table 10. TFLite variants of the feature extractor (test n = 130, single run)

Variant	Size (MB)	Accuracy	Weighted F1	Δ vs float	Edge extraction, mean (ms)
Float32 (reference)	≈1.55 (computed)	89.23%	0.8924	–	–
float16	0.783	89.23%	0.8924	0.00	5.25
Dynamic-range (8-bit weights)	0.418	88.46%	0.8848	−0.77	8.32
Full-integer INT8 (100 rep. images)	0.437	80.00%	0.7993	−9.23	7.91
Full-integer INT8 (300 rep. images)	0.437	80.00%	0.8022	−9.23	7.98

Dynamic-range quantization affected only Coccidiosis recall (0.8684 to 0.8421) and Healthy precision (0.8205 to 0.8000), whereas full-integer INT8 lowered the F1-score of all classes, most for Salmonella (0.8986 to 0.7869).

The 74.9% payload reduction does not depend on quantization because the feature vector is sent as float32. The latency in Table 12 was measured with an earlier TFLite conversion whose accuracy could not be reproduced; re-measuring the reconverted models gave 5.2–8.3 ms edge extraction (Table 10), so the Table 12 values are kept as the more conservative figures.

Table 11 lists published figures for context only, because datasets and protocols differ (training on farm- and laboratory-labeled images and testing on farm-labeled images [5], 10,500 images [1], and live chickens rather than droppings [15]); it is not a controlled benchmark and no superiority is claimed. The proposed model reports lower accuracy than these figures but is far smaller (0.42 MB in TFLite dynamic-range form and about 1.55 MB in float32, versus 26–98 MB).

Table 11. Quantitative Comparison with Existing Models

Model	Size (MB)	Parameters	Accuracy	Dataset and protocol	Same data as this study?
MobileNetV2 [5]	26 MB	3.4 Million	98.02%	Same repository [15]; trained on farm- and laboratory-labeled images, tested on farm-labeled images	Partly (same repository, different training and test data)
ResNet50 [1]	~98 MB	25.6 Million	98.7%	10,500 chicken dropping images	Partly (larger subset of the same repository)
YOLOv5 & ResNet50 [15]	Not reported (standard ResNet50: ≈98 MB)	Not reported (standard ResNet50: 25.6 Million)	99%	Images of live chickens (not droppings); ResNet50 trained on 5,000 images (source not specified), 80/20 train/test split, 100 epochs	No (different task and image source)
Proposed (Hybrid)	0.42 MB (TFLite); ≈1.55 MB (float32)	0.39 Million	89.23%	1,255 laboratory-labeled images, 130-image test set	Yes

4.6.1. Measured Latency (Laptop-Class CPU Emulating the Edge Role)

End-to-end latency was measured across the 130 test images using a Hugging Face cloud endpoint over a standard (non-cellular) connection, with feature extraction on the laptop-class device (Table 12).

Table 12. Measured Latency Components (Laptop-Class CPU Emulating the Edge Role, n=130, Single Run)

Component	Mean (ms)	Median (ms)	P95 (ms)	Min (ms)	Max (ms)
Edge Feature Extraction	27.79	27.30	50.83	13.46	125.71
Serialization	0.04	0.02	0.05	0.02	2.33
Network (client ↔ cloud)	1342.22	1309.56	1506.09	1176.19	1906.80
Cloud Classification (Trio-Boosting)	7.30	7.00	8.37	6.19	15.74
Total End-to-End	**1377.35**	**1341.16**	**1535.57**	**1201.40**	**1938.76**

Network transmission accounted for 97.4% of the 1377.35 ms end-to-end latency despite the 2.0 KB payload, driven mainly by shared-infrastructure overhead (cold start, gateway routing, request queues), so it is not representative of real 2G/3G/4G transmission times.

A cloud-only baseline sending the raw image (8,161 bytes), measured in a separate session, gave a mean of 1554.60 ms, so the split-inference mean (1377.35 ms) was 11.4% lower. Because platform latency varies widely (1201.40–1938.76 ms) and the sessions were not interleaved, this difference is not evidence of a latency advantage.

4.6.2. Estimated Transmission Time Under Assumed Network Conditions (Analytical)

Table 13 gives analytical estimates (not measurements) of transmission time, $T = S \times 8 / B + RTT$ (Eq. 13), where S is the measured payload (8,161 B raw, 2,048 B feature vector), B is the assumed bandwidth, and RTT is the midpoint of the assumed range; computation time is excluded.

Table 13. Estimated (Analytical) Transmission Time Under Assumed Network Conditions (Not Measured)

Network	Bandwidth	RTT (range; midpoint used)	Raw image (8,161 B)	Feature vector (2,048 B)	Reduction
2G (GPRS/EDGE)	~50 Kbps	300–600 ms (450)	1755.8 ms	777.7 ms	55.7%
3G (UMTS/HSPA)	384 Kbps	~100 ms	270.0 ms	142.7 ms	47.2%
4G/LTE	~10 Mbps	20–50 ms (35)	41.5 ms	36.6 ms	11.8%

The estimated reduction falls from 55.7% (2G) to 11.8% (4G/LTE) because the round-trip time, not the payload, dominates at higher bandwidth; at 4G/LTE the difference (about 4.9 ms) is smaller than the assumed RTT range of 20–50 ms.

4.7. Discussion and Key Findings

The ablation results suggest that the pooling strategy contributed more substantially to classification performance than the choice of classifier. GCP outperformed GAP by 22.31 percentage points and GMP by 3.08 percentage points, whereas Weighted Soft Voting achieved the same accuracy as the best individual classifier and did not demonstrate a clear advantage over simpler classifiers (Tables 4–6). These findings suggest that the quality of the extracted feature representation may be more influential than increasing classifier complexity within the evaluated experimental setting.

Figure 7 presents representative prediction results for the four classes, with the images originally available at 96 × 96 pixels and subsequently upsampled for model input. The measured model-side computational latency was 35.09 ms, compared with the 25.8 ms inference time reported in prior work [5]. However, these values are not directly comparable because the hardware, measurement scope, and experimental conditions differ. Moreover, the present measurement was obtained using a laptop-class CPU to emulate the edge-side device. It therefore indicates that the model-side computation is relatively lightweight under the tested

conditions, but does not establish that real-time performance can be achieved on resource-constrained edge devices. The measured end-to-end latency of approximately 1.38 s is of a similar order of magnitude to the roughly 3 s reported for a cloud-hosted telediagnosis application [18], although differences in tasks, payloads, and platforms preclude a direct quantitative comparison.



Figure 7. Detection Results: (**a**) Healthy; (**b**) Coccidiosis; (**c**) NCD; (**d**) Salmonella

The results are consistent with the rationale for split inference, in which extracted features are transmitted instead of the original image [44]. Nevertheless, the estimated transmission-time reduction decreases from 55.7% under the assumed 2G conditions to 11.8% under 4G/LTE conditions (Table 13), while the measured end-to-end latency does not establish a clear overall latency advantage. This distinction highlights that reducing the transmitted data volume does not necessarily produce a comparable reduction in total response time, particularly when network and shared-infrastructure overheads remain substantial. The baseline comparison also indicates that the proposed model's principal advantage lies in its compact size rather than superior classification accuracy (Table 7).

Among the four classes, NCD was the easiest to identify, achieving the highest class-wise F1-score. One possible explanation is that its dark-green coloration contrasts with the light brown or gray background, providing more visually distinctive features. The Grad-CAM visualizations in Fig. 8 are consistent with this interpretation: attention is more concentrated on the greenish regions in the illustrated NCD example, whereas the Coccidiosis example exhibits more diffuse activation [45]. These visualizations provide qualitative insight into the model's attention patterns, although they do not independently establish that the highlighted regions are uniquely associated with the corresponding diseases.

The principal classification difficulty concerns the Healthy and Coccidiosis classes, as illustrated in Fig. 9. Five Coccidiosis images were misclassified as Healthy, compared with one Healthy image misclassified as Coccidiosis. This asymmetry is potentially important in practical screening because missed infections may delay intervention and increase the risk of disease spread. One possible explanation is that early-stage Coccidiosis may exhibit faint blood spots that resemble natural feed-pigment variations in healthy droppings. However, this interpretation should be treated cautiously because six test images belonged to duplicate

pairs with conflicting labels, suggesting that source-label inconsistencies may also have contributed to the observed errors (Section 4.8.1).

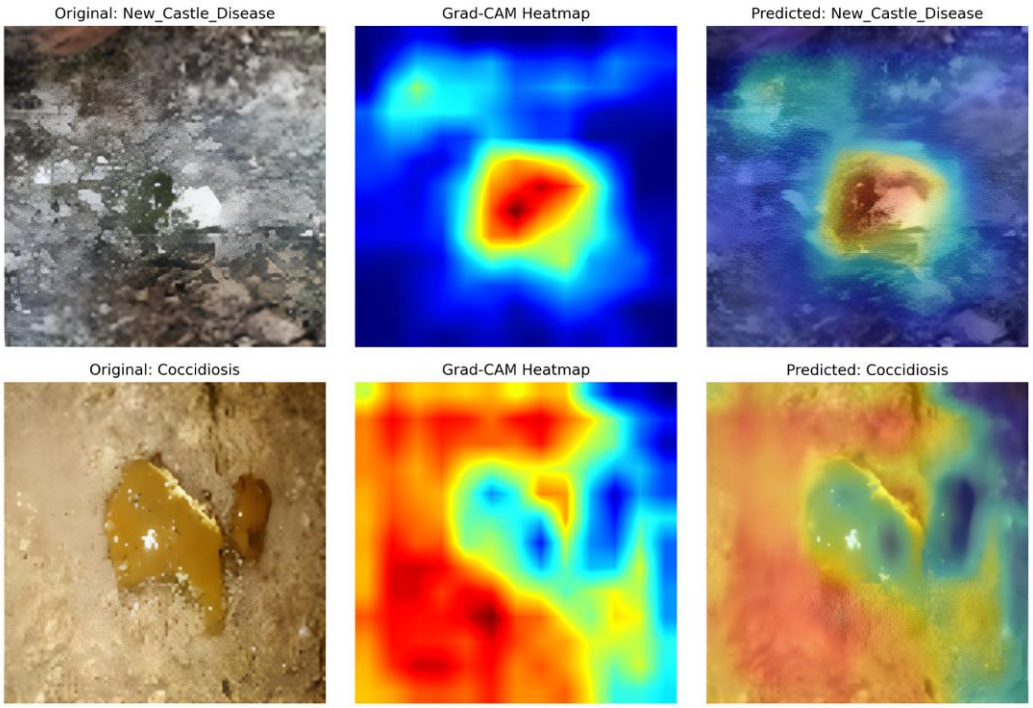


Figure 8. Grad-CAM visualization



Figure 9. Similar Characteristic

The results indicate that simple classifiers performed comparably to the Trio-Boosting ensemble on the evaluated dataset, although ensemble learning has been reported to improve medical image classification in other contexts [39]. The proposed system achieved approximately 88–89% accuracy in the float and dynamic-range quantized configurations while maintaining a compact model and reducing the amount of data transmitted per inference. These characteristics support the potential of resource-efficient deep learning as a direction for AI-assisted poultry disease screening in resource-constrained livestock environments [2], [28]. However, the reported accuracy should be interpreted within the scope of the available dataset rather than as an estimate of performance on unseen farms. The test images were drawn from the same repository as the training data, and previous work has reported lower multiclass

classification performance—approximately 0.69–0.70—on data collected in Malawi [4]. Although differences in datasets and evaluation protocols prevent a direct comparison, this evidence underscores the importance of evaluating generalization across geographic regions and acquisition conditions. Further validation using independently collected field data and direct testing on constrained edge devices is therefore needed before broader deployment claims can be made.

4.8. Limitations and Threats to Validity

The findings should be interpreted in light of limitations related to dataset quality, deployment validation, and experimental design. These limitations affect the generalizability and reproducibility of the reported results.

4.8.1. Dataset and Preprocessing Limitations

The dataset comprises 1,255 images from a single public repository and one laboratory-labeled subset. Its size and provenance limit the diversity of visual conditions represented, particularly in terms of lighting, cameras, farm environments, and poultry breeds. These constraints are relevant to visually similar classes, particularly Healthy and Coccidiosis, and mean that generalization beyond the evaluated data remains preliminary [29]. In addition, the source images were originally downsampled to 96 × 96 pixels and subsequently upsampled to 224 × 224 pixels for model input. Although this preprocessing provides the required input dimensions, it cannot recover fine-grained information lost during downsampling and may reduce the availability of subtle texture cues. This concern is consistent with findings that image compression can remove clinically relevant visual information [3].

The evaluation used a single fixed data split, without an independently collected test set or farm-level partitioning. Consequently, the results do not establish performance on images from previously unseen farms. The NCD class also contains only 20 test images, making its class-specific metrics sensitive to individual classification outcomes.

A post hoc duplicate check identified exact duplicates across data partitions: 9 of the 130 test images and 9 of the 123 validation images had duplicates in another partition. Excluding the nine affected test images yielded an accuracy of 89.26% and a weighted F1-score of 0.8926 on the remaining 121 images, compared with 89.23% and 0.8924, respectively, on the full test set (Table 9). These results suggest that the identified cross-partition duplicates had little effect on the aggregate metrics, although other sources of dataset bias remain.

More importantly, 19 of the 35 duplicate pairs contained identical images with conflicting labels: 16 Coccidiosis–Healthy pairs and three Healthy–Salmonella pairs. Six test images belonged to such pairs, including an identical image assigned to both Healthy and Coccidiosis within the test set. Because identical images with conflicting labels cannot both be classified correctly by a deterministic classifier, these inconsistencies complicate the interpretation of class-specific errors. Accordingly, some of the Healthy–Coccidiosis confusion discussed in Section 4.7 and illustrated in Figure 9 may reflect annotation inconsistencies in the source data, rather than model behavior alone. These images were retained in the reported evaluation, and the findings should be interpreted with this limitation in mind.

4.8.2. Hardware and Network Validation

Deployment-related findings are subject to limitations in hardware and network validation. Latency and quantization were evaluated using a laptop-class CPU to emulate the edge role rather than resource-constrained devices such as the Raspberry Pi, ESP32, or Jetson Nano. Accordingly, performance, power consumption, and memory requirements on such devices remain unverified under field conditions [1]. The transmission times estimated for 2G, 3G, and 4G/LTE conditions (Table 13) were calculated from nominal network specifications rather than measured on actual cellular networks. Real-world latency may also be affected by cloud-side factors, including cold starts, request queues, and shared infrastructure [3]. In addition, the cloud-only baseline and split-inference approach were measured in separate sessions rather than in interleaved runs, meaning that differences in end-to-end latency may partly reflect variations in platform conditions.

The latency measurements also require qualification in relation to model conversion. The edge-side latency reported in Table 12 was obtained using an earlier TensorFlow Lite (TFLite) conversion whose accuracy could not be reproduced in subsequent conversions (Table 10). Therefore, the latency and quantized-accuracy results should be interpreted separately

rather than as a verified performance profile of a single, reproducible model variant. Further testing on actual edge devices and under measured network conditions is necessary to establish the practical deployment benefits of the proposed approach.

4.8.3. Ablation Design, Ensemble Contribution, and Statistical Uncertainty

The experimental limitations discussed in this subsection complement the dataset and deployment constraints described in Sections 4.8.1 and 4.8.2. Table 14 summarizes these limitations, their supporting evidence, and the corresponding claims that require cautious interpretation.

Table 14. Summary of limitations and the claims they restrict

Limitation	Evidence in this study	Restricted claim
Small, single-repository dataset	1,255 images; one laboratory-labeled subset; 20 NCD test images	Generalization to other farms, cameras, and lighting
Duplicates and conflicting labels	9 test and 9 validation duplicates; 19 of 35 duplicate pairs with different labels	Healthy–Coccidiosis error analysis
Single split, single run	130 test images; 95% CI of 82.7–93.5%	Ranking of models differing by a few images
Single-run final model versus retrained runs	Table 7 and Table 8: retrained backbone 83.59 ± 2.35% versus final model 89.23% (single run)	Expected accuracy of the architecture; any claim of accuracy parity with ResNet50
Emulated edge hardware	Laptop-class CPU; no power or memory measurement	Suitability for constrained devices
Shared cloud endpoint	1.38 s end-to-end; 97.5% network	Any real-time claim
Analytical network estimates	Table 13 assumptions; no cellular measurement	Latency under 2G/3G/4G
Staged ablation, ensemble gain	Additive design; ensemble equals CatBoost	Independent contribution of each component

The ablation experiments used early stopping with a minimum of 15 epochs, whereas the final model was trained for 100 epochs. Consequently, the best ablation accuracy of 85.38% and the final accuracy of 89.23% are not directly comparable; the difference may reflect both training duration and run-to-run variation. The staged, additive ablation design also does not isolate the independent contribution of each component. A leave-one-component-out evaluation would provide stronger evidence of the individual contributions and remains a direction for future work.

CatBoost alone achieved the same accuracy as the Trio-Boosting ensemble (85.38%), so the added value of Weighted Soft Voting was not established. Although ensemble methods have been explored in related contexts [39], the current results do not demonstrate that the ensemble consistently improves performance over a simpler classifier. This consideration is relevant when selecting hybrid architectures for datasets of comparable size. Similarly, the simple classifiers evaluated using the final model's features were within six test images of Trio-Boosting (Table 6). With only 130 test images, the McNemar test has limited power to distinguish modest performance differences; therefore, these comparisons do not establish statistical equivalence among the classifiers.

The final model correctly classified 116 of the 130 test images, corresponding to an accuracy of 89.23% and a 95% Wilson confidence interval of 82.7–93.5%. With this test-set size, one additional or fewer correct predictions changes accuracy by approximately 0.77 percentage points. The overlapping confidence intervals in Table 6 further indicate that small differences between classifiers should be interpreted cautiously. These intervals do not account for training variability or all effects of duplicated and inconsistently labeled images.

The final model's accuracy also represents a single training run, whereas the retrained backbone achieved 83.59 ± 2.35% across three random seeds (Tables 7 and 8). This difference highlights the uncertainty associated with relying on a single run and means that 89.23% should be interpreted as the observed performance of the selected model, rather than the expected accuracy of the architecture across repeated training runs. In addition, the validation

set informed checkpoint selection, learning-rate scheduling, and model-weight selection, while baseline hyperparameters were also tuned during experimentation. The TFLite variant was selected after test accuracy had been compared; consequently, its reported accuracy of 88.46% should not be regarded as an unbiased estimate from an untouched test set.

Taken together, these findings support the observed performance of the selected model under the evaluated conditions, but do not establish consistent gains from ensembling or the expected performance of the architecture across repeated runs. Further experiments using repeated training, more comprehensive component ablations, and an independent test set would strengthen the reliability of these conclusions.

5. Conclusions

This study evaluated a lightweight SqueezeNet-based feature extractor incorporating GCP and Trio-Boosting within a split-inference edge–cloud framework for poultry disease classification. The results show that GCP was the primary contributor to classification performance, whereas Trio-Boosting provided no statistically significant improvement over simpler classifiers. Under a matched evaluation protocol, the retrained backbone achieved an accuracy of 83.59 ±2.35%, comparable to fine-tuned MobileNetV2 but below fine-tuned ResNet50. With approximately 0.39 million parameters, the proposed backbone offers a compact alternative, although its advantage lies in model efficiency rather than superior predictive accuracy. The separately trained final model achieved 89.23% accuracy, but this result was not consistently reproduced during retraining.

Class-level evaluation showed that Newcastle disease (NCD) achieved the highest F1-score (0.9524), whereas Healthy and Coccidiosis were more difficult to distinguish, with F1-scores of 0.8533 and 0.8919, respectively. Conflicting labels in the source dataset may partly explain this confusion, although their individual effects were not independently established. Regarding communication efficiency, split inference reduced the transmitted payload by 74.9% compared with raw-image transmission. However, end-to-end latency (1,377.35 ms) substantially exceeded model-side latency (35.09 ms), indicating that communication and infrastructure overhead remained important bottlenecks. The dynamic-range TFLite model retained 88.46% accuracy at a model size of 0.42 MB, whereas full-integer INT8 quantization reduced accuracy to 80.00%.

Overall, the findings demonstrate the potential of combining compact feature extraction with split inference to reduce transmission requirements, but do not establish superior classification accuracy or deployment readiness. The small, single-source dataset and emulated hardware and network conditions limit generalizability. Future work should prioritize external validation across farms and acquisition conditions [31]. Future work should also include testing on physical edge devices under real network conditions, systematic component ablation, and evaluation across a broader range of poultry diseases.

Author Contributions: Conceptualization: Y.S.; Methodology: Y.S.; Software: Y.S.; Validation: E.U. and A.A.; Formal analysis: Y.S.; Investigation: E.U., A.A. and S.R.; Resources: Y.S.; Data curation: E.U. and A.A.; Writing—original draft preparation: Y.S.; Writing—review and editing: E.U., A.A. and S.R.; Visualization: Y.S.; Supervision: E.U., A.A. and S.R.; Project administration: E.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: The data supporting the reported results are publicly available in Zenodo Repository [29].

Acknowledgments: The authors would like to express their deepest gratitude and appreciation to Universitas Amikom Yogyakarta for providing the facilities, infrastructure, and a supportive research environment throughout the completion of this study. The authors would also like to thank their colleagues at the Faculty of Computer Science for their productive discussions and technical assistance. Furthermore, the use of AI tools (Google AI Studio) was limited in scope and was used to assist in refining the linguistic structure of this manuscript.

Conflicts of Interest: The authors declare that there are no conflicts of interest related to this study. The entire research process, from data collection to interpretation of results, was conducted objectively without any influence from external parties or commercial relationships. Universitas Amikom Yogyakarta, as the affiliated institution, played no role in the research study; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results of this study.

References

[1] M. Z. Degu and G. L. Simegn, "Smartphone based detection and classification of poultry diseases from chicken fecal images using deep learning techniques," *Smart Agric. Technol.*, vol. 4, p. 100221, Aug. 2023, doi: 10.1016/j.atech.2023.100221.

[2] M. Naeem *et al.*, "Advancements in machine learning applications in poultry farming: a literature review," *J. Appl. Poult. Res.*, vol. 34, no. 4, p. 100602, Dec. 2025, doi: 10.1016/j.japr.2025.100602.

[3] A. Dhungana, X. Yang, B. Paneru, S. Dahal, G. Lu, and L. Chai, "An Integrated Deep Learning Approach for Poultry Disease Detection and Classification Based on Analysis of Chicken Manure Images," *AgriEngineering*, vol. 7, no. 9, p. 278, Aug. 2025, doi: 10.3390/agriengineering7090278.

[4] G. Chidziwisano, E. Samikwa, and C. Daka, "Deep learning methods for poultry disease prediction using images," *Comput. Electron. Agric.*, vol. 230, p. 109765, Mar. 2025, doi: 10.1016/j.compag.2024.109765.

[5] D. Machuve, E. Nwankwo, N. Mduma, and J. Mbelwa, "Poultry diseases diagnostics models using deep learning," *Front. Artif. Intell.*, vol. 5, Aug. 2022, doi: 10.3389/frai.2022.733345.

[6] H. Zhu *et al.*, "Intelligent agriculture: deep learning in UAV-based remote sensing imagery for crop diseases and pests detection," *Front. Plant Sci.*, vol. 15, Oct. 2024, doi: 10.3389/fpls.2024.1435016.

[7] R. Chin, C. Catal, and A. Kassahun, "Plant disease detection using drones in precision agriculture," *Precis. Agric.*, vol. 24, no. 5, pp. 1663–1682, Oct. 2023, doi: 10.1007/s11119-023-10014-y.

[8] P. Joshi, M. Hasanuzzaman, C. Thapa, H. Afli, and T. Scully, "Enabling All In-Edge Deep Learning: A Literature Review," *IEEE Access*, vol. 11, pp. 3431–3460, 2023, doi: 10.1109/ACCESS.2023.3234761.

[9] D. Ngo, H.-C. Park, and B. Kang, "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments," *Electronics*, vol. 14, no. 12, p. 2495, Jun. 2025, doi: 10.3390/electronics14122495.

[10] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review," *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, Jan. 2023, doi: 10.1109/JPROC.2022.3226481.

[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv*. Nov. 04, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[12] L. S. Bernardo, R. Damaševičius, S. H. Ling, V. H. C. de Albuquerque, and J. M. R. S. Tavares, "Modified SqueezeNet Architecture for Parkinson's Disease Detection Based on Keypress Data," *Biomedicines*, vol. 10, no. 11, p. 2746, Oct. 2022, doi: 10.3390/biomedicines10112746.

[13] H. Li, X. Yue, and L. Meng, "Enhanced mechanisms of pooling and channel attention for deep learning feature maps," *PeerJ Comput. Sci.*, vol. 8, p. e1161, Nov. 2022, doi: 10.7717/peerj-cs.1161.

[14] N. R. Pratama, D. R. I. M. Setiadi, I. Harkespan, and A. A. Ojugo, "Feature Fusion with Albumentation for Enhancing Monkeypox Detection Using Deep Learning Models," *J. Comput. Theor. Appl.*, vol. 2, no. 3, pp. 427–440, Feb. 2025, doi: 10.62411/jcta.12255.

[15] W. Ali, I. Ud Din, A. Almogren, and J. J. P. C. Rodrigues, "Poultry Health Monitoring With Advanced Imaging: Toward Next-Generation Agricultural Applications in Consumer Electronics," *IEEE Trans. Consum. Electron.*, vol. 70, no. 4, pp. 7147–7154, Nov. 2024, doi: 10.1109/TCE.2024.3409069.

[16] S. Wang *et al.*, "Advances in Deep Learning Applications for Plant Disease and Pest Detection: A Review," *Remote Sens.*, vol. 17, no. 4, p. 698, Feb. 2025, doi: 10.3390/rs17040698.

[17] G. Latif, S. E. Abdelhamid, R. E. Mallouhy, J. Alghazo, and Z. A. Kazimi, "Deep Learning Utilization in Agriculture: Detection of Rice Plant Diseases Using an Improved CNN Model," *Plants*, vol. 11, no. 17, p. 2230, Aug. 2022, doi: 10.3390/plants11172230.

[18] M. T. H. Khan Tusar, M. T. Islam, A. H. Sakil, M. N. H. N. Khandaker, and M. M. Hossain, "An Intelligent Telediagnosis of Acute Lymphoblastic Leukemia using Histopathological Deep Learning," *J. Comput. Theor. Appl.*, vol. 2, no. 1, pp. 1–12, May 2024, doi: 10.62411/jcta.10358.

[19] L. Zhao and Z. Zhang, "A improved pooling method for convolutional neural networks," *Sci. Rep.*, vol. 14, no. 1, p. 1589, Jan. 2024, doi: 10.1038/s41598-024-51258-6.

[20] H. Xu and M. Cheng, "Regularized Optimal Transport Layers for Generalized Global Pooling Operations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 12, pp. 15426–15444, Dec. 2023, doi: 10.1109/TPAMI.2023.3314661.

[21] R. Azadnia, M. M. Al-Amidi, H. Mohammadi, M. A. Cifci, A. Daryab, and E. Cavallo, "An AI Based Approach for Medicinal Plant Identification Using Deep CNN Based on Global Average Pooling," *Agronomy*, vol. 12, no. 11, p. 2723, Nov. 2022, doi: 10.3390/agronomy12112723.

[22] A. K. M. Shoriful Islam, M. Rakib Hassan, M. Uddin, and M. S. Rahman, "Handcrafted feature-based poultry disease detection from fecal images," *Smart Agric. Technol.*, vol. 12, p. 101557, Dec. 2025, doi: 10.1016/j.atech.2025.101557.

[23] Z. Lin *et al.*, "Efficient Parallel Split Learning Over Resource-Constrained Wireless Edge Networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 10, pp. 9224–9239, Oct. 2024, doi: 10.1109/TMC.2024.3359040.

[24] Y. Chen, T. Luo, W. Fang, and N. N. Xiong, "EdgeCI: Distributed Workload Assignment and Model Partitioning for CNN Inference on Edge Clusters," *ACM Trans. Internet Technol.*, vol. 24, no. 2, pp. 1–24, May 2024, doi: 10.1145/3656041.

[25] T. Liu *et al.*, "Spatial Channel Attention for Deep Convolutional Neural Networks," *Mathematics*, vol. 10, no. 10, p. 1750, May 2022, doi: 10.3390/math10101750.
[26] A. M. Alayba, E. M. Senan, and J. S. Alshudukhi, "Enhancing early detection of Alzheimer's disease through hybrid models based on feature fusion of multi-CNN and handcrafted features," *Sci. Rep.*, vol. 14, no. 1, p. 31203, Dec. 2024, doi: 10.1038/s41598-024-82544-y.
[27] K. P. Zaw and A. Mon, "Enhanced Multi-Class Skin Lesion Classification of Dermoscopic Images Using an Ensemble of Deep Learning Models," *J. Comput. Theor. Appl.*, vol. 2, no. 2, pp. 256–267, Nov. 2024, doi: 10.62411/jcta.11530.
[28] M. H. Natsir, W. F. Mahmudy, M. Tono, and Y. F. Nuningtyas, "Advancements in artificial intelligence and machine learning for poultry farming: Applications, challenges, and future prospects," *Smart Agric. Technol.*, vol. 12, p. 101307, Dec. 2025, doi: 10.1016/j.atech.2025.101307.
[29] D. Machuve, E. Nwankwo, N. Mduma, H. Mbelwa, E. Maguo, and C. Munisi, "Machine Learning Dataset for Poultry Diseases Diagnostics," *Zenodo*. Zenodo, 2021. doi: 10.5281/zenodo.4495536.
[30] R. E. Ako *et al.*, "Effects of Data Resampling on Predicting Customer Churn via a Comparative Tree-based Random Forest and XGBoost," *J. Comput. Theor. Appl.*, vol. 2, no. 1, pp. 86–101, Jun. 2024, doi: 10.62411/jcta.10562.
[31] A. Priyambodo, R. R. Isnanto, and R. Sanjaya, "A Systematic Literature Review of Robustness-Aware Batik Motif Classification: Acquisition Variability, Feature Representation, and Learning Models," *J. Comput. Theor. Appl.*, vol. 4, no. 1, pp. 21–42, Jun. 2026, doi: 10.62411/jcta.16074.
[32] A. Zafar *et al.*, "Convolutional Neural Networks: A Comprehensive Evaluation and Benchmarking of Pooling Layer Variants," *Symmetry (Basel).*, vol. 16, no. 11, p. 1516, Nov. 2024, doi: 10.3390/sym16111516.
[33] D. Guo, Z. Wu, J. Feng, and T. Zou, "Multi-scale semantic enhancement network for object detection," *Sci. Rep.*, vol. 13, no. 1, p. 7178, May 2023, doi: 10.1038/s41598-023-34277-7.
[34] Z. UrRehman *et al.*, "Effective lung nodule detection using deep CNN with dual attention mechanisms," *Sci. Rep.*, vol. 14, no. 1, p. 3934, Feb. 2024, doi: 10.1038/s41598-024-51833-x.
[35] M. R. Abbasniya, S. A. Sheikholeslamzadeh, H. Nasiri, and S. Emami, "Classification of Breast Tumors Based on Histopathology Images Using Deep Features and Ensemble of Gradient Boosting Methods," *Comput. Electr. Eng.*, vol. 103, no. 1, p. 108382, Jan. 2022, doi: 10.1016/j.compeleceng.2022.108382.
[36] T. Chen and C. Guestrin, "XGBoost," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[37] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[38] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, Jan. 2018. [Online]. Available: http://arxiv.org/abs/1706.09516
[39] D. Muller, I. Soto-Rey, and F. Kramer, "An Analysis on Ensemble Learning Optimized Medical Image Classification With Deep Convolutional Neural Networks," *IEEE Access*, vol. 10, pp. 66467–66480, 2022, doi: 10.1109/ACCESS.2022.3182399.
[40] Q. McNemar, "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947, doi: 10.1007/BF02295996.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, vol. 2016-Decem, pp. 770–778. doi: 10.1109/CVPR.2016.90.
[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.
[43] E. B. Wilson, "Probable Inference, the Law of Succession, and Statistical Inference," *J. Am. Stat. Assoc.*, vol. 22, no. 158, pp. 209–212, Jun. 1927, doi: 10.1080/01621459.1927.10502953.
[44] M. Li, X. Zhang, J. Guo, and F. Li, "Cloud–Edge Collaborative Inference with Network Pruning," *Electronics*, vol. 12, no. 17, p. 3598, Aug. 2023, doi: 10.3390/electronics12173598.
[45] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, vol. 128, no. 2, pp. 618–626. doi: 10.1109/ICCV.2017.74.