Transfer Learning-Based Classification of poultry Diseases for Enhanced Health Management

Team ID:

LTVIP2025TMID37405

Team Members:

- Karrothu Thapaswi
- Joga Bhanu Prakash
- Polaki Chandana
- Yadla Harika

Phase-1: Brainstroming &Ideation

Objective

Problem Statement:

Poultry farming is a critical livelihood in many communities, but poultry diseases can spread quickly, resulting in significant economic losses. Access to veterinary care is limited in rural regions, making early diagnosis difficult. Manual diagnosis based on symptoms is error-prone and time-consuming. **Proposed solution**:

This project proposes a **deep learning-based system using transfer learning** to classify poultry diseases into four categories:

- Salmonella
- New Castle Disease

- Coccidiosis
- Healthy

By integrating this model into a mobile/web app, farmers can receive instant disease predictions and treatment suggestions based on input data such as symptoms, environmental conditions, and biological images.

Purpose of the Project:

- Empower farmers with an AI-based diagnostic tool.
- Promote early detection and containment of poultry diseases.
- Reduce dependence on expert veterinarians in remote areas.
- Improve poultry productivity and minimize losses.

Impact of the Project

- **Early Detection** of diseases to avoid outbreaks.
- **Instant Diagnosis** using mobile input (symptoms/images).
- **Education & Awareness** among farmers and vet students.
- **Automation** in poultry health management and reduced manual dependency.

Phase 2: Requirements Analysis

Functional Requirements

- Upload symptom data and environmental observations.
- Optionally upload images of infected birds or feces.
- Display predicted disease class with confidence score.
- Provide suggested treatment and management practices.
- User-friendly interface for rural usability.

Technical Requirements

- **Languages:** Python, JavaScript
- **Frameworks:** TensorFlow/Keras, Flask or Django (backend), React or HTML/CSS (frontend)
- **Tools:** Jupyter Notebook, Google Colab, OpenCV

Constraints & Challenges

- Dataset size and quality.
- Image variability (lighting, background).
- Deployment compatibility across devices (web/mobile).
- Model generalization to real-world inputs.

Phase 3: Project Design

System Architecture

User Flow

1. Open app
2. Input symptoms/data or upload image
3. Click "Predict"
4. System returns disease class and suggestions

UI/UX Considerations

- Simple input forms with dropdowns/images
- Language localization (English + regional)
- Offline support (optional with local storage)

Phase 4: Project Planning (Agile Methodology)

Sprint Planning

- Dataset collection: poultry images, symptoms
- Model selection and training (Transfer Learning with ResNet50 or MobileNet)
- Backend API for predictions
- Frontend interface for farmers
- Testing and bug fixes
- Final report & deployment

Task Allocation

- **You**: Model training, integration
- **Teammate**: UI/UX design, documentation, deployment

Timeline & Milestones

-Data preprocessing, initial model training

-Model tuning and UI prototyping

-Backend + frontend integration

-Testing, feedback collection

-Final adjustments and deployment

Phase 5: Project Development

Technology Stack

- Python, HTML/CSS/JavaScript
- Flask or Django backend
- TensorFlow/Keras for model
- Pre-trained CNN: ResNet50/MobileNet
- .h5 model file exported
- Google Colab or Jupyter for model dev

Development Steps

1. Preprocess dataset (labeling, resizing)
2. Train model using transfer learning

3. Export model
4. Build Flask backend (app.py)
5. Design UI (HTML templates or React)
6. Connect frontend to backend
7. Deploy (Heroku, Render, or local server)

Phase 6: Testing & Validation

Test Scenarios

- Input symptom data → Predicts disease accurately
- Input image → Detects correct class
- Invalid input → Shows appropriate error
- Test on mobile + desktop
- Multi-disease test scenarios

Bug Fixes

- Model not responding → Optimized Flask routing
- Incorrect predictions → Improved data preprocessing

UI glitches on mobile → Applied responsive design

Deployment

- Hosted using Flask backend
- Local server or cloud (e.g., Heroku, Render)

Folder structure:

Real-World Scenarios

Scenario 1: Rural Community Outbreak

- Input symptoms: lethargy, diarrhea
- Model predicts **Coccidiosis**
- App suggests treatment → farmers act quickly → spread contained

Scenario 2: Commercial Farm Monitoring

- Daily health checks via app
- Early detection of **New Castle Disease**
- Quarantine initiated → major outbreak prevented

Scenario 3: Veterinary Training

- Vet students simulate real-world diagnosis
- Learn AI-aided diagnostic workflow
- Better equipped to handle field cases