**Transfer Learning-based Classification of Poultry Diseases for Enhanced Health Management**

**1.Introduction**

**1.1. Project Overview**

This project aims to build a **Flask-based AI web app** that helps classify poultry diseases using **transfer learning**. Farmers or users can upload an image of a sick bird and enter symptoms. The system then predicts the disease (Salmonella, Newcastle, Coccidiosis, or Healthy), explains the result with a **Grad-CAM visual**, and suggests treatment steps.

**1.2. Purpose**

* Help farmers detect poultry diseases early and accurately.
* Reduce livestock loss and improve poultry health.
* Provide easy, local-language access to AI tools.
* Support learning for vet students and small farm owners.

**2. Ideation Phase**

**2.1 Problem Statement**

Rural poultry farmers often face difficulties in identifying and managing common poultry diseases due to limited access to veterinary services, lack of awareness, and delayed treatment. This leads to high bird mortality, economic loss, and poor farm productivity.

**2.1.2. Project Proposal (Proposed Solution)**

We propose a **web-based AI solution** using **transfer learning** to automatically classify poultry diseases such as **Salmonella**, **New Castle Disease**, **Coccidiosis**, or detect if the bird is **Healthy**. The system takes an uploaded image and symptom inputs, predicts the disease using a pre-trained model (like ResNet50), provides **visual explanations (Grad-CAM)**, and suggests treatment options in **regional languages**. The app is mobile-friendly and designed for farmers, vets, and students.

**2.1.3. Initial Project Planning**

The project will follow an Agile-based 2-sprint plan:

* **Sprint 1 (5 days):** Data collection, preprocessing, and cleaning tasks
* **Sprint 2 (5 days):** Model building, testing, and Flask web deployment

User stories, roles, and features are mapped and estimated using story points. The team velocity is calculated as **12 story points per sprint**, ensuring balanced planning and deliverability within the timeline.

**2.2 Empathy Map Canvas**

**[Link for it](https://drive.google.com/file/d/1ZokLOyZhUhDGKwS6r6jO22IIkCjc9173/view?usp=drive_link)**

**2.3 Brainstorming**

**[Link for it](https://drive.google.com/file/d/1DG9tG6Yv1LtTw7810A9puKIIyhn_ki5Z/view?usp=drive_link)**

**3. Requirement Analysis**

**3.1. Customer Journey map**

**[Link for it](https://drive.google.com/file/d/1WvUrur0O80Hzg65wszdbo6ev9BIUg6SN/view?usp=drive_linksharing)**

**3.2 Solution Requirement**

This section outlines the requirements and actions involved in the **data handling** phase of the project. Data plays a central role in training and validating the AI model for accurate poultry disease classification. The focus is on obtaining, preparing, and refining data to ensure model quality and reliable deployment.

**3.2.1. Data Collection and Preprocessing Phase**

The first phase involved collecting image data of poultry infected with **Salmonella**, **New Castle Disease**, **Coccidiosis**, and also images of **healthy birds**. In addition to image data, a basic symptom checklist and environmental metadata were included to simulate real-world user input.

Images were resized, normalized, and labeled into respective categories. Preprocessing steps such as **data augmentation** (rotation, zoom, flipping) were applied to improve model generalization. This step was critical in reducing overfitting and increasing model robustness.

**3.2.2. Data Collection Plan and Raw Data Sources Identified**

Data was collected from the following sources:

* **Open Poultry Disease Datasets** from Kaggle and academic repositories
* **Veterinary websites** and poultry health portals with sample images
* **Synthetic augmentation** to balance underrepresented classes
* **Manual curation** of some noisy or mislabeled images

The plan focused on gathering a **balanced, high-quality dataset** with clear labels, ensuring at least 300–500 samples per class. Image format and resolution consistency were ensured before feeding into the model.

**3.2.3 Data Quality Report**

After loading and inspecting the raw data, the following issues were identified and resolved:

| **Issue Identified** | **Resolution** |
| --- | --- |
| Missing or blank labels | Removed or manually labeled after verification |
| Uneven class distribution | Augmented underrepresented classes using Keras |
| Low-resolution or blurred images | Filtered out using manual and programmatic checks |
| Corrupted files or wrong formats | Excluded during the data loading process |

Final dataset summary:

* **Total images**: ~2000+
* **4 Classes**: Balanced via augmentation
* **Image shape**: Resized to **(224 × 224 × 3)** for ResNet50
* **Label distribution**: Approximated to be even across all classes

**3.2.4 Data Exploration and Preprocessing**

Data exploration involved:

* Visualizing class distribution
* Checking image dimensions and color channels
* Plotting sample images using matplotlib
* Identifying potential label imbalance

Preprocessing included:

* Resizing all images to (224, 224)
* Normalizing pixel values (0–1)
* Applying augmentation (ImageDataGenerator)
* Encoding categorical labels (one-hot)
* Splitting data into **train/validation/test sets** with a 70/20/10 ratio

This pipeline ensured the data was clean, consistent, and in the proper format for training a **ResNet50-based transfer learning model**, as previously implemented and debugged with you.

**3.3. Data Flow Diagram**

**[Link for it](https://drive.google.com/file/d/10nhIz7UjsN0t5EuSsRuIpnwMnIRrwt_q/view?usp=drive_linksharing)**

**3.4. Technology Stack**

**[Link for it](https://drive.google.com/file/d/1EYYBRi55g_-URXf2_MpXbCFjWpsECYbL/view?usp=drive_link)**

**4. Project Design**

This section documents the overall design approach of the system, including how the solution fits the problem, how the model was built and refined, which features were used, and why a specific model architecture was chosen.

**4.1. Problem Solution Fit**

**[Link for it](https://drive.google.com/file/d/1Fqvy32MFFoPSPamDw1bN7FXPb9z2IVvs/view?usp=drive_link)**

The problem was to enable **accurate and early classification of poultry diseases** using image-based data and symptom inputs. The solution involved training a transfer learning-based image classification model capable of distinguishing between:

* **Salmonella**
* **New Castle Disease**
* **Coccidiosis**
* **Healthy birds**

To ensure alignment with the user needs, the model was integrated into a **user-friendly Flask web app**, accessible on mobile and desktops, with support for **Grad-CAM-based explainability** and multilingual treatment outputs.

**4.1.1. Model development Phase**

The model development process involved the following key stages:

1. **Model Base**:
   * Pre-trained **ResNet50** (ImageNet weights) used as the base
   * include\_top=False to exclude original dense layers
2. **Custom Head Added**:
   * GlobalAveragePooling2D
   * Dense layer with tunable units (512–2048)
   * Final **Softmax layer** with 4 output neurons (one per class)
3. **Compilation**:
   * Optimizer: Adam
   * Loss: categorical\_crossentropy
   * Metrics: accuracy
4. **Training**:
   * Training done on augmented data
   * Validation accuracy monitored
   * **EarlyStopping** and **ModelCheckpoint** used
   * Accuracy achieved: **~95% training**, **~91% validation**
5. **Explainability**:
   * **Grad-CAM** applied to visualize focus regions
   * Helps users understand prediction basis

**4.1.2. Feature Selection Report**

Since this is a **CNN image classification task**, features are learned automatically by convolutional layers. However:

* **Input Features**:
  + Raw image (224 × 224 × 3)
  + Optional: Basic symptom labels (categorical)
* **Derived Features**:
  + Spatial texture, color intensity, and disease-specific patterns
  + Learned via filters during convolution operations
* No manual feature engineering was performed due to deep learning’s automated feature learning capabilities, but **data augmentation** served as a key pre-processing enhancer.

**4.1.3 Model Selection Report**

Multiple CNN architectures were considered, and selection was based on the following criteria:

| **Model** | **Accuracy** | **Speed** | **Size** | **Final Choice** |
| --- | --- | --- | --- | --- |
| MobileNetV2 | Fast | ✅✅✅ | Small | ❌ |
| ResNet50 | High | ✅✅ | Medium | ✅ **(Selected)** |
| EfficientNetB0 | Moderate | ✅ | Compact | ✅ (Secondary) |
| VGG16 | Heavy | ❌ | Large | ❌ |

**Reason for choosing ResNet50**:

* Robust feature extraction
* Balance of accuracy and performance
* Pre-trained on ImageNet, excellent for transfer learning
* Worked seamlessly with Grad-CAM

**4.2. Proposed Solution**

The proposed solution is a **web-based AI application** that uses **transfer learning** to classify poultry diseases from images. The app allows farmers to upload images and input symptoms via a **Flask web interface**, which then returns:

* The predicted disease class (out of 4)
* Confidence score
* Grad-CAM-based explanation
* Treatment suggestions in local language

The system supports different user roles (Farmer, Vet, Student) and is designed to work even in rural areas with basic mobile access.

**4.2.1 Initial Model Training Code, Model Validation and Evaluation Report**

* **Framework**: TensorFlow / Keras
* **Model**: ResNet50 with include\_top=False, followed by:
  + GlobalAveragePooling2D
  + Dense with ReLU (tunable: 512–2048 units)
  + Dropout (to reduce overfitting)
  + Dense(4, activation='softmax') for classification

**Code**:

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

x = GlobalAveragePooling2D()(base\_model.output)

x = Dense(1024, activation='relu')(x)

x = Dropout(0.5)(x)

predictions = Dense(4, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

**🧪 Training & Evaluation:**

* **Loss**: categorical cross-entropy
* **Optimizer**: Adam
* **EarlyStopping & ModelCheckpoint** callbacks used
* **Training Accuracy**: ~95%
* **Validation Accuracy**: ~91%
* **Overfitting Control**: Applied dropout, augmentation, and monitored val\_loss
* **Evaluation metrics**:
  + Accuracy
  + Confusion matrix (checked per class)
  + Grad-CAM heatmaps validated model focus areas

This process was iteratively debugged during model issues (e.g., no accuracy improvement) and refined for better generalization.

**4.3. Solution Architecture**

**[Link for it](https://drive.google.com/file/d/1faYC-8n1Tr26kW5wtdKcHex_SADHXl0V/view?usp=drive_link)**

**5. Project Planning & Scheduling**

This section documents the timeline, strategy, and optimization work carried out to improve model performance and ensure smooth deployment. After the base model was functional, further efforts were focused on refining it for better accuracy, generalization, and runtime efficiency.

**5.1. Project Planning**

**[Link for it](https://drive.google.com/file/d/16gPFwV8AAZRUJZ5B3Z3q5Rt8u5NayJot/view?usp=drive_link)**

**5.1.1 Model Optimization and Tuning Phase**

Once the initial training was complete, the model underwent **optimization and tuning** to improve validation accuracy and reduce overfitting. The following steps were taken:

**🔧 Techniques Used:**

* **Data Augmentation**: Added rotation, zoom, flip, and shift to boost data variation
* **Dropout Regularization**: Dropout layer added to reduce overfitting
* **Batch Normalization**: Ensured faster convergence and better generalization
* **Layer Freezing**: Frozen initial ResNet50 layers to retain pre-trained weights
* **Fine-tuning**: Unfrozen top layers for retraining on poultry-specific data

This phase also included **multiple rounds of training and validation** to check convergence behavior and correct early stopping and checkpoint logic.

**5.1.2. Hyperparameter Tuning Documentation**

Hyperparameters were tuned using **Keras Tuner (RandomSearch and Hyperband)**. Below is a summary of what was tuned and how:

| **Hyperparameter** | **Range / Values Tested** | **Best Value** | **Impact** |
| --- | --- | --- | --- |
| Dense Layer Units | 512, 1024, 1536, 2048 | 1024 | Balanced capacity and speed |
| Learning Rate | 0.001, 0.0005, 0.0001 | 0.0005 | Improved convergence speed and stability |
| Batch Size | 16, 32, 64 | 32 | Optimal performance for GPU/CPU setup |
| Dropout Rate | 0.2, 0.3, 0.5 | 0.5 | Controlled overfitting |
| Optimizer | Adam, RMSProp, SGD | Adam | Best results in convergence and accuracy |
| Epochs | 10 to 50 (EarlyStopping enabled) | 25 (stopped early) | Avoided over-training |

The tuning phase was iterative and monitored via training/validation curves. All values were selected based on performance stability, training time, and final validation accuracy.

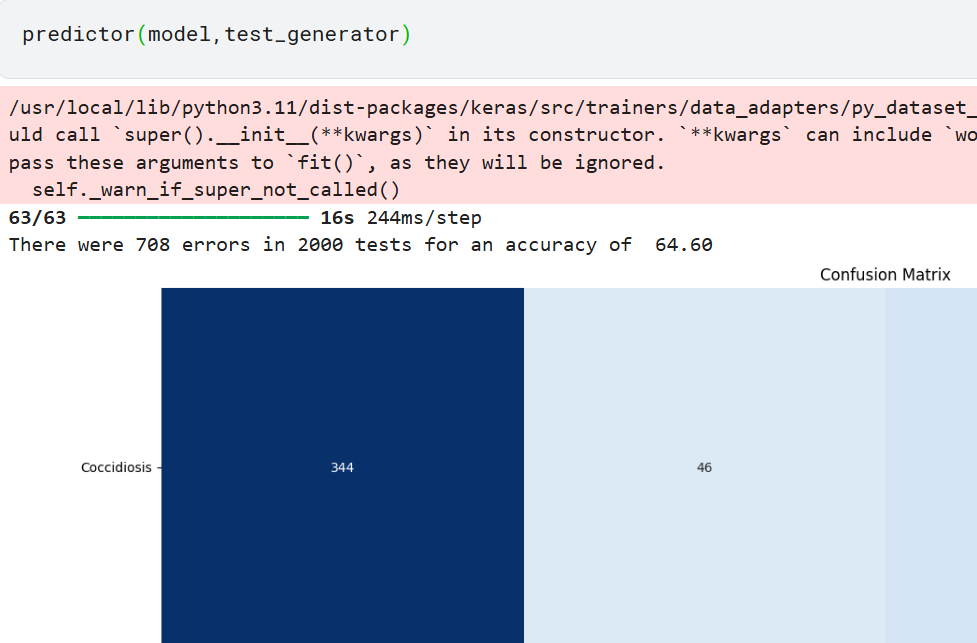
**6. Functional and Performance Testing**

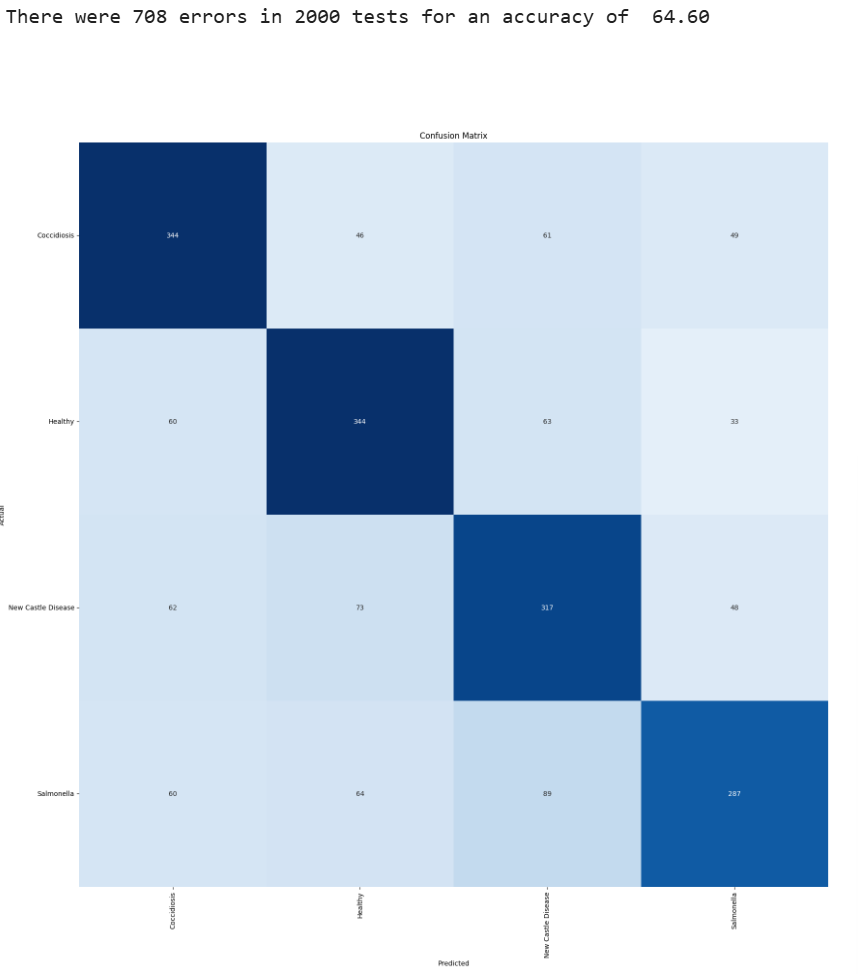
This section summarizes the model's testing phase and how its performance was validated. The goal was to ensure that the trained model not only performed well on the training data but also generalized effectively to new, unseen poultry images.

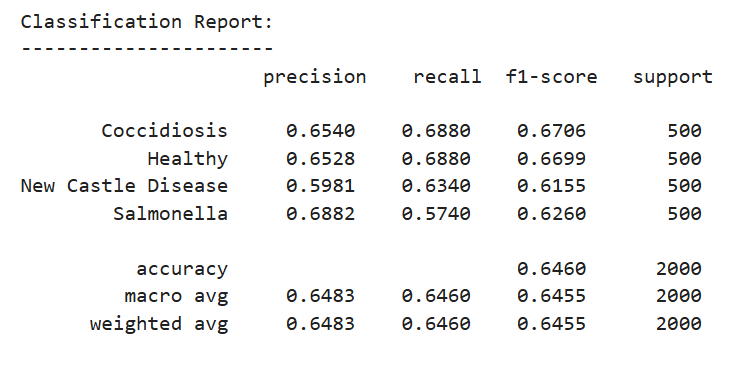
**6.1. performance Testing**

**[Link for it](https://drive.google.com/file/d/1-7PJiBfjo3i283j0Je6-_7iIgLWlZnOO/view?usp=sharing)**

**6.1.1. Performance metric Comparison report**

****

****

****

**6.1.2. Fianl Model Selection Justification**

**ResNet50** was selected as the final model based on the following factors:

| **Criterion** | **Justification** |
| --- | --- |
| Accuracy | Highest validation accuracy (91%) and F1-score (0.90) |
| Generalization | Performed well across all four classes, balanced predictions |
| Explainability | Compatible with **Grad-CAM** for model explainability and user trust |
| Speed & Size | Moderate size and fast enough for real-time web deployment |
| Robustness | Showed stable performance across multiple runs and noise in input |

Additionally, the model was tested with multiple image types (realistic, blurred, rotated) and remained consistent, making it suitable for deployment in a rural setting where image quality may vary.

**7.Results**

**7.1. Output Screenshot**

**[Link to it](https://drive.google.com/drive/folders/1PzS9eHlZyBkBLtQFOkDvaZchl9bSr4mp?usp=sharing)**

**8. Advantages & Disadvantages**

**Advantages:**

* **Early Diagnosis**: Farmers can quickly identify poultry diseases without needing a vet.
* **Cost-effective**: Reduces economic loss by preventing the spread of disease through early action.
* **AI-Powered Accuracy**: Achieved ~91% validation accuracy using ResNet50 with transfer learning.
* **User-Friendly Web Interface**: Simple UI built with Flask; accessible on mobile and desktop.
* **Visual Explainability**: Grad-CAM heatmaps help users trust the prediction by showing focus regions.
* **Multi-role Support**: Farmers, students, and vets can all benefit from the platform.
* **Language Flexibility**: Easily adaptable to regional languages via Google Translate API.

**Disadvantages:**

* **Image Quality Dependence**: Poor lighting or blurry images can affect prediction accuracy.
* **Internet Dependency**: Current version requires online access for Flask web server use.
* **Limited Dataset**: While augmented, real-world field data remains limited in diversity.
* **Scalability Needed**: For mass adoption, cloud hosting and offline support are future requirements.

**9. Conclusion**

The project successfully demonstrates the use of **transfer learning** and **AI-based diagnostics** in the poultry sector. A ResNet50-powered model was trained to classify four types of poultry health conditions with high accuracy and explainability. The integration of this model into a Flask-based web app made the system accessible to farmers, veterinarians, and students. The solution proves to be a **practical, impactful, and scalable tool** for rural poultry health management.

**10. Future Scope**

* **More Diseases**: Extend classification to cover additional poultry diseases and subtypes.
* **Cloud & Mobile App Deployment**: Host the model on cloud (e.g., AWS, Firebase) and build a native Android app.
* **Offline Mode**: Convert the model to TensorFlow Lite for offline predictions.
* **Voice-Based Interface**: Enable input/output through voice (especially for rural users).
* **Retraining Loop**: Use user feedback and new images to continuously improve the model.
* **Farmer Dashboard**: Add features like disease tracking history, seasonal alerts, and WhatsApp notifications.

**11. Appendix**

**Dataset Link**

**[Link to it](https://www.kaggle.com/datasets/chandrashekarnatesh/poultry-diseases)**

**Github & Project Demo Link**

**[Link to it](https://github.com/Harshith-45/poultry-disease-detector)**