## **Transfer learning baced classification of poultry diseases for enhanced health management**

**🎯Objective:**

To build a computer vision model using Transfer Learning to classify poultry diseases based on images of infected birds, supporting early detection and improved health management in poultry farms.

**🎯Architecture:**

1️⃣ Input

* Images of poultry (sick or healthy)
* Stored in labeled folders (e.g., Fowlpox/, Healthy/, etc.)

2️⃣ Preprocessing

* Resize images to 224x224 pixels
* Normalize pixel values
* Data Augmentation (flip, rotate, zoom)

3️⃣ Transfer Learning Model

* Use MobileNetV2 (or ResNet) as a base model
* Freeze base layers (pre-trained on ImageNet)
* Add custom layers:

4️⃣ Training

* Train the model on preprocessed images
* Monitor validation accuracy
* Save the best model as poultry\_model.h5

5️⃣ Prediction

* Load the saved model
* Input a new image
* Output: Predicted disease class with confidence %

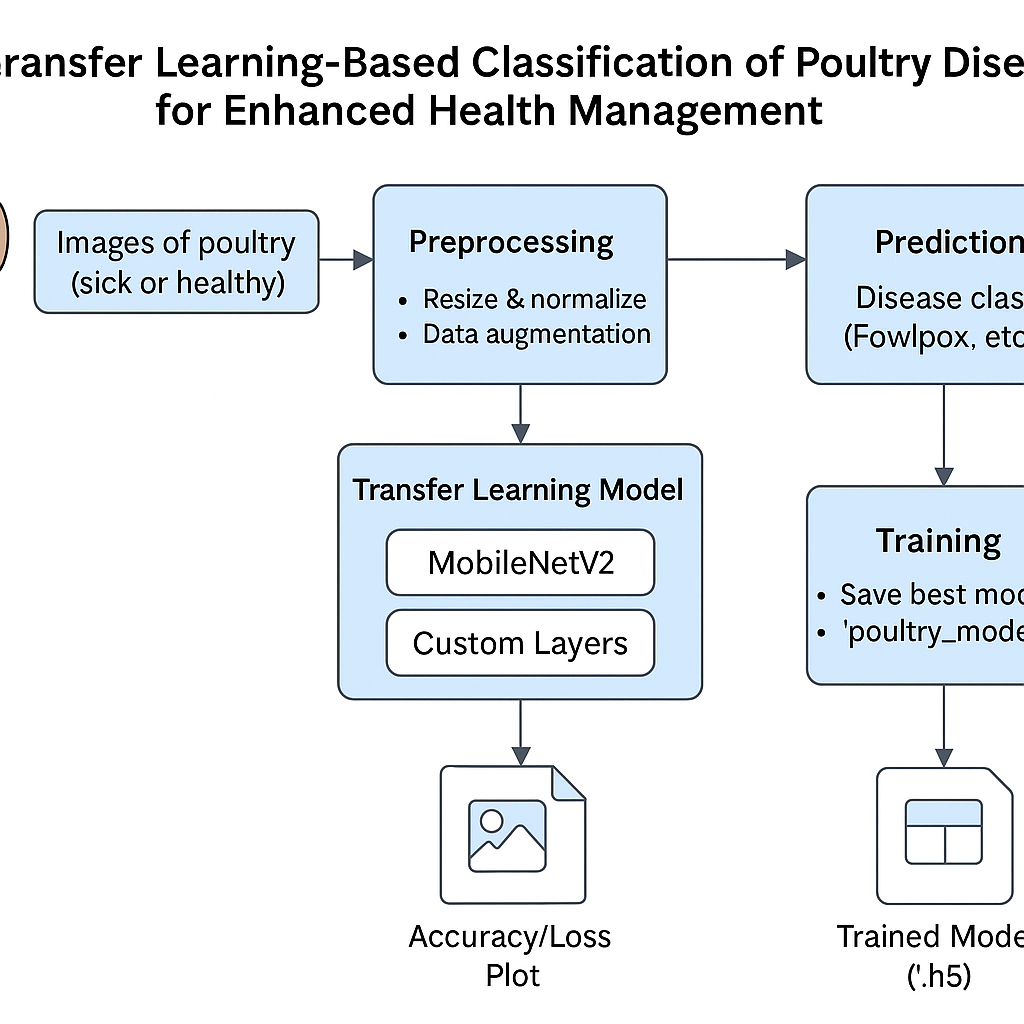
6️⃣ Outputs

* Trained Model (.h5)
* Accuracy/Loss Plot (.png)
* Disease prediction from input images

**🔧 Tools Used:**

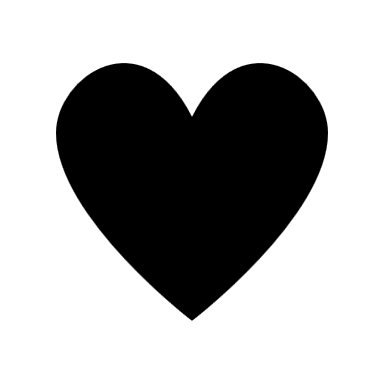
* Python
* TensorFlow/Keras
* Matplotlib
* VS Code

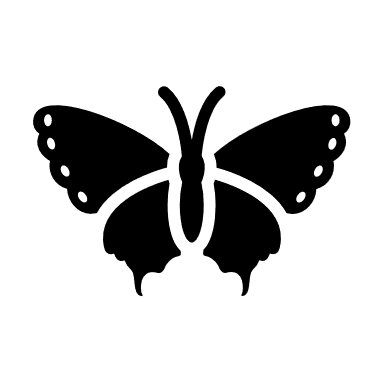
Structure:

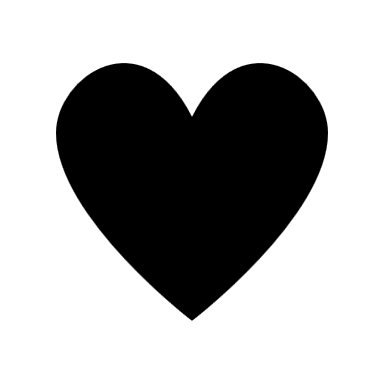


**Pre requisites:**

Software requirements:

Python 3.7+

Vs code

Pip(python package manager)

**Project structure:**

poultry\_disease\_project/

├── dataset/

│ ├── Fowlpox/

│ ├── Coccidiosis/

│ ├── Newcastle/

│ ├── AvianInfluenza/

│ ├── InfectiousBronchitis/

│ └── Healthy/

├── model/

│ └── poultry\_model.h5 ← Saved after training

├── src/

│ ├── train\_model.py ← Core model training code

│ └── predict\_image.py ← Optional inference script

├── requirements.txt

└── README.md

**Data collection and preparation:**

📦 1. Data Collection

🔍 A. What You Need

Images of poultry showing visible symptoms of diseases and healthy birds.

**Source:**

Kaggle

Github repositories

Veterinary websites

Collaboration with poultry farms

**ImageDataGenerator:**

It's a TensorFlow/Keras tool that:

* Loads images from folders
* Applies real-time data augmentation
* Rescales and resizes images automatically
* Splits data into training and validation

**Testing model and data prediction:**

import numpy as np

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import os

# Load trained model

model\_path = os.path.join('..', 'model', 'poultry\_model.h5')

model = load\_model(model\_path)

# Define class names (should match your training classes)

class\_names = ['AvianInfluenza', 'Coccidiosis', 'Fowlpox', 'Healthy', 'InfectiousBronchitis', 'Newcastle']

# Load test image

img\_path = os.path.join('..', 'test\_images', 'sample.jpg') # Replace with your image name

img = image.load\_img(img\_path, target\_size=(224, 224)) # Resize same as training

img\_array = image.img\_to\_array(img) / 255.0 # Normalize

img\_array = np.expand\_dims(img\_array, axis=0) # Add batch dimension

# Predict

pred = model.predict(img\_array)

class\_idx = np.argmax(pred)

confidence = pred[0][class\_idx]

print(f"\nPredicted class: {class\_names[class\_idx]}")

print(f"Confidence: {confidence \* 100:.2f}%")

**how to use:**

save the above code

place a test image

run from terminal;

**output:**

opens a window showing the test image

title include predicted disease;

**Application Building:**

We will use **streamlit**

Stream lit:

Python:

import streamlit as st

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

from PIL import Image

# Load the model

model = load\_model('../model/poultry\_model.h5')

# Class names from your training labels

class\_names = ['AvianInfluenza', 'Coccidiosis', 'Fowlpox', 'Healthy', 'InfectiousBronchitis', 'Newcastle']

# Streamlit app UI

st.set\_page\_config(page\_title="Poultry Disease Classifier")

st.title("🐔 Poultry Disease Detection App")

st.write("Upload a poultry image to predict the disease class.")

uploaded\_file = st.file\_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

if uploaded\_file is not None:

# Load and display image

img = Image.open(uploaded\_file)

st.image(img, caption='Uploaded Image', use\_column\_width=True)

# Preprocess image

img = img.resize((224, 224))

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

# Predict

predictions = model.predict(img\_array)

pred\_class = class\_names[np.argmax(predictions)]

confidence = np.max(predictions) \* 100

# Display prediction

st.success(f"✅ Prediction: \*\*{pred\_class}\*\*")

st.info(f"Confidence: \*\*{confidence:.2f}%\*\*")

**🖼️ Output:**

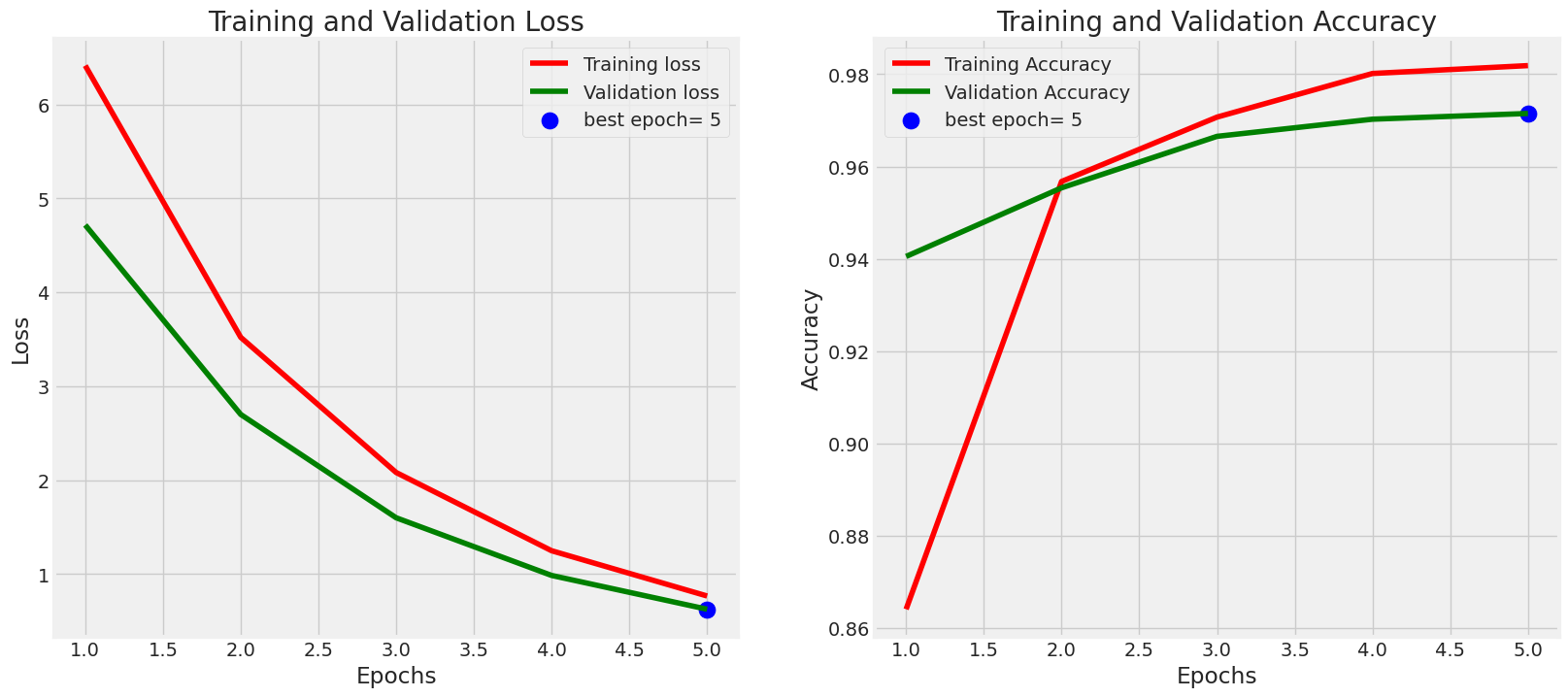
You’ll see:

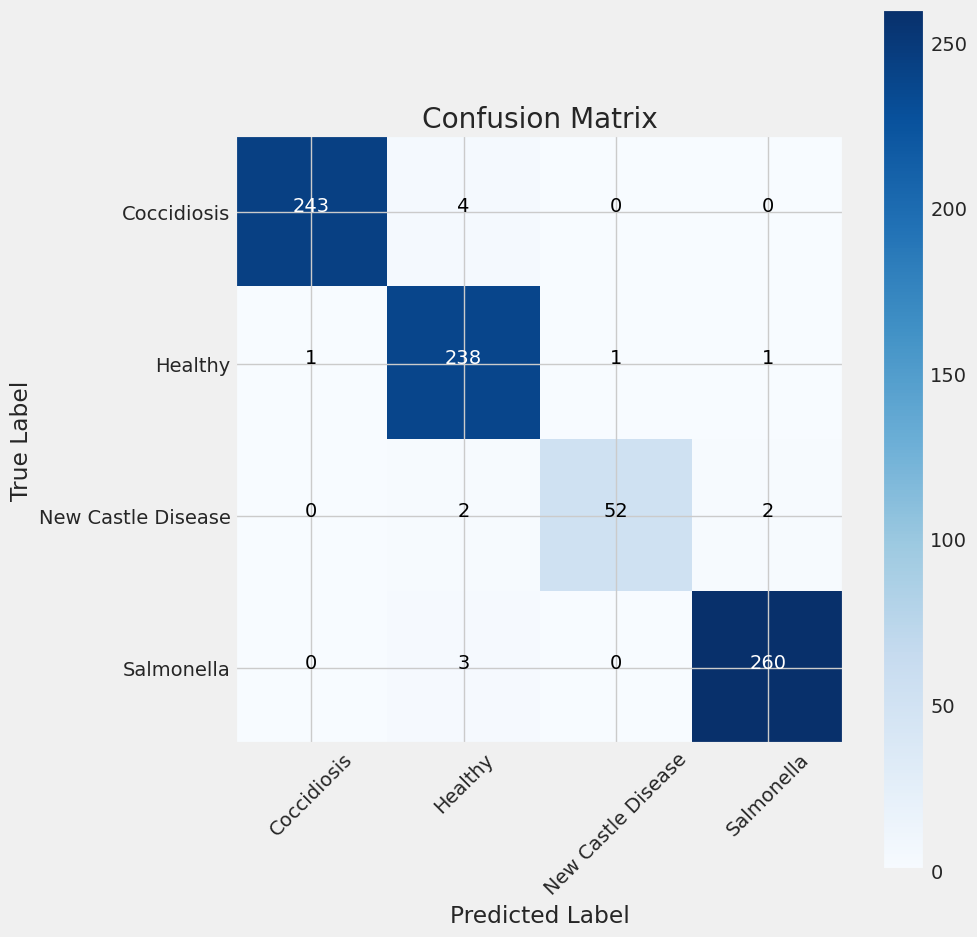
* Image preview
* Predicted disease class
* Confidence score

**Example:**

****

**Plot training(history):**



**Sample prediction:**