# Real-time Cotton Disease Detection with Raspberry Pi & Streamlit

Here's a complete solution for real-time cotton disease detection using Raspberry Pi 4 with camera module and Streamlit UI:

## 1. Raspberry Pi Camera Setup Code (`pi\_camera\_detection.py`)

```python

import cv2

import numpy as np

import tensorflow as tf

from PIL import Image

import time

import requests

import io

import json

# Load the trained model

model = tf.keras.models.load\_model('cotton\_disease\_model.h5')

with open('class\_names.json', 'r') as f:

class\_names = json.load(f)

# Disease information database

disease\_info = {

'Aphids': {

'sprinkle': True,

'advice\_en': 'Use insecticidal soap or neem oil. Introduce natural predators like ladybugs.',

'advice\_hi': 'कीटनाशक साबुन या नीम का तेल उपयोग करें। लेडीबग जैसे प्राकृतिक शिकारियों को पेश करें।',

'pesticide\_en': 'Imidacloprid, Acetamiprid, or Thiamethoxam',

'pesticide\_hi': 'इमिडाक्लोप्रिड, एसिटामिप्रिड, या थायमेथोक्सम'

},

'Army worm': {

'sprinkle': True,

'advice\_en': 'Handpick worms or use biological controls like Bacillus thuringiensis (Bt).',

'advice\_hi': 'हाथ से इल्लियों को उठाएं या बैसिलस थुरिंजिएन्सिस (Bt) जैसे जैविक नियंत्रण का उपयोग करें।',

'pesticide\_en': 'Chlorantraniliprole, Spinosad, or Emamectin benzoate',

'pesticide\_hi': 'क्लोरान्ट्रानिलिप्रोल, स्पिनोसैड, या इमामेक्टिन बेंजोएट'

},

'Bacterial blight': {

'sprinkle': True,

'advice\_en': 'Remove infected plants. Use copper-based bactericides and practice crop rotation.',

'advice\_hi': 'संक्रमित पौधों को हटा दें। तांबा-आधारित जीवाणुनाशक का उपयोग करें और फसल चक्रण का अभ्यास करें।',

'pesticide\_en': 'Copper oxychloride, Streptomycin, or Kasugamycin',

'pesticide\_hi': 'कॉपर ऑक्सीक्लोराइड, स्ट्रेप्टोमाइसिन, या कासुगामाइसिन'

},

'Cotton Boll Rot': {

'sprinkle': True,

'advice\_en': 'Improve air circulation. Avoid overhead irrigation. Apply fungicides during flowering.',

'advice\_hi': 'हवा के संचार में सुधार करें। ऊपरी सिंचाई से बचें। फूल आने के दौरान कवकनाशी लगाएं।',

'pesticide\_en': 'Carbendazim, Mancozeb, or Propiconazole',

'pesticide\_hi': 'कार्बेन्डाजिम, मैंकोजेब, या प्रोपिकोनाजोल'

},

'Green Cotton Boll': {

'sprinkle': False,

'advice\_en': 'No treatment needed. Maintain good agricultural practices.',

'advice\_hi': 'किसी उपचार की आवश्यकता नहीं है। अच्छी कृषि पद्धतियों को बनाए रखें।',

'pesticide\_en': 'None required',

'pesticide\_hi': 'आवश्यकता नहीं'

},

'Healthy': {

'sprinkle': False,

'advice\_en': 'No treatment needed. Continue with regular monitoring and good agricultural practices.',

'advice\_hi': 'किसी उपचार की आवश्यकता नहीं है। नियमित निगरानी और अच्छी कृषि पद्धतियों को जारी रखें।',

'pesticide\_en': 'None required',

'pesticide\_hi': 'आवश्यकता नहीं'

},

'Powdery mildew': {

'sprinkle': True,

'advice\_en': 'Improve air circulation. Apply sulfur-based or systemic fungicides.',

'advice\_hi': 'हवा के संचार में सुधार करें। गंधक-आधारित या प्रणालीगत कवकनाशी लगाएं।',

'pesticide\_en': 'Sulfur, Myclobutanil, or Tebuconazole',

'pesticide\_hi': 'सल्फर, माइक्लोब्यूटानिल, या टेबुकोनाजोल'

},

'Target spot': {

'sprinkle': True,

'advice\_en': 'Remove infected leaves. Apply fungicides and practice crop rotation.',

'advice\_hi': 'संक्रमित पत्तियों को हटा दें। कवकनाशी लगाएं और फसल चक्रण का अभ्यास करें।',

'pesticide\_en': 'Chlorothalonil, Azoxystrobin, or Pyraclostrobin',

'pesticide\_hi': 'क्लोरोथैलोनिल, एज़ोक्सिस्ट्रोबिन, या पाइराक्लोस्ट्रोबिन'

}

}

def preprocess\_image(image):

"""Preprocess image for model prediction"""

image = image.resize((224, 224))

image\_array = np.array(image) / 255.0

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

return image\_array

def predict\_disease(image):

"""Predict disease from image"""

processed\_image = preprocess\_image(image)

predictions = model.predict(processed\_image, verbose=0)

predicted\_class\_idx = np.argmax(predictions[0])

predicted\_class = class\_names[predicted\_class\_idx]

confidence = round(100 \* np.max(predictions[0]), 2)

return predicted\_class, confidence

def capture\_and\_predict():

"""Capture image from camera and predict disease"""

# Initialize camera (for Raspberry Pi camera module)

camera = cv2.VideoCapture(0)

# Set camera resolution

camera.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 640)

camera.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 480)

print("Camera initialized. Press 'c' to capture, 'q' to quit.")

while True:

ret, frame = camera.read()

if not ret:

print("Failed to capture image")

break

# Display the frame

cv2.imshow('Cotton Disease Detection - Press c to capture', frame)

key = cv2.waitKey(1) & 0xFF

if key == ord('c'):

# Convert BGR to RGB

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

pil\_image = Image.fromarray(rgb\_frame)

# Predict disease

predicted\_class, confidence = predict\_disease(pil\_image)

# Get disease information

disease\_data = disease\_info.get(predicted\_class, {})

# Prepare result

result = {

'disease': predicted\_class,

'confidence': confidence,

'sprinkle': disease\_data.get('sprinkle', False),

'advice\_en': disease\_data.get('advice\_en', ''),

'advice\_hi': disease\_data.get('advice\_hi', ''),

'pesticide\_en': disease\_data.get('pesticide\_en', ''),

'pesticide\_hi': disease\_data.get('pesticide\_hi', ''),

'timestamp': time.strftime("%Y-%m-%d %H:%M:%S")

}

print(f"Predicted: {predicted\_class} ({confidence}%)")

print(f"Sprinkle: {'Yes' if result['sprinkle'] else 'No'}")

# Send to Streamlit server (if running)

try:

response = requests.post('http://localhost:8502/api/prediction', json=result)

if response.status\_code == 200:

print("Result sent to Streamlit UI")

except:

print("Streamlit server not available")

elif key == ord('q'):

break

camera.release()

cv2.destroyAllWindows()

if \_\_name\_\_ == "\_\_main\_\_":

capture\_and\_predict()

```

## 2. Streamlit Web Interface (`streamlit\_app.py`)

```python

import streamlit as st

import requests

import json

import time

from datetime import datetime

import pandas as pd

from PIL import Image

import io

import base64

# Page configuration

st.set\_page\_config(

page\_title="Smart Cotton Doctor",

page\_icon="🌿",

layout="wide",

initial\_sidebar\_state="expanded"

)

# Custom CSS

st.markdown("""

<style>

.main-header {

font-size: 3rem;

color: #2e7d32;

text-align: center;

margin-bottom: 2rem;

}

.sub-header {

font-size: 1.5rem;

color: #388e3c;

margin-bottom: 1rem;

}

.result-card {

background-color: #f1f8e9;

padding: 2rem;

border-radius: 10px;

margin-bottom: 2rem;

border-left: 5px solid #4caf50;

}

.disease-card {

background-color: #ffebee;

padding: 2rem;

border-radius: 10px;

margin-bottom: 2rem;

border-left: 5px solid #f44336;

}

.healthy-card {

background-color: #e8f5e8;

padding: 2rem;

border-radius: 10px;

margin-bottom: 2rem;

border-left: 5px solid #4caf50;

}

.sprinkle-yes {

color: #f44336;

font-weight: bold;

font-size: 1.2rem;

}

.sprinkle-no {

color: #4caf50;

font-weight: bold;

font-size: 1.2rem;

}

</style>

""", unsafe\_allow\_html=True)

# Disease information database (same as in camera script)

disease\_info = {

'Aphids': {

'sprinkle': True,

'advice\_en': 'Small sap-sucking insects that weaken plants. They cause curling and yellowing of leaves.',

'advice\_hi': 'छोटे कीट जो पौधों का रस चूसते हैं और पौधों को कमजोर करते हैं। वे पत्तियों के मुड़ने और पीले पड़ने का कारण बनते हैं।',

'pesticide\_en': 'Imidacloprid, Acetamiprid, or Thiamethoxam',

'pesticide\_hi': 'इमिडाक्लोप्रिड, एसिटामिप्रिड, या थायमेथोक्सम',

'severity': 'Moderate'

},

'Army worm': {

'sprinkle': True,

'advice\_en': 'Caterpillars that feed on leaves and can defoliate plants completely.',

'advice\_hi': 'इल्लियाँ जो पत्तियों को खाती हैं और पौधों को पूरी तरह से पत्तीविहीन कर सकती हैं।',

'pesticide\_en': 'Chlorantraniliprole, Spinosad, or Emamectin benzoate',

'pesticide\_hi': 'क्लोरान्ट्रानिलिप्रोल, स्पिनोसैड, या इमामेक्टिन बेंजोएट',

'severity': 'High'

},

'Bacterial blight': {

'sprinkle': True,

'advice\_en': 'Bacterial disease causing water-soaked lesions that turn brown and angular leaf spots.',

'advice\_hi': 'जीवाणु जनित रोग जो पानी से भरे घाव पैदा करता है जो भूरे हो जाते हैं और कोणीय पत्ती के धब्बे पैदा करते हैं।',

'pesticide\_en': 'Copper oxychloride, Streptomycin, or Kasugamycin',

'pesticide\_hi': 'कॉपर ऑक्सीक्लोराइड, स्ट्रेप्टोमाइसिन, या कासुगामाइसिन',

'severity': 'High'

},

'Cotton Boll Rot': {

'sprinkle': True,

'advice\_en': 'Fungal disease causing bolls to rot and turn black. Favored by humid conditions.',

'advice\_hi': 'फफूंदी जनित रोग जो टिंडों को सड़ने और काला करने का कारण बनता है। आर्द्र परिस्थितियों में फलता-फूलता है।',

'pesticide\_en': 'Carbendazim, Mancozeb, or Propiconazole',

'pesticide\_hi': 'कार्बेन्डाजिम, मैंकोजेब, या प्रोपिकोनाजोल',

'severity': 'Moderate'

},

'Green Cotton Boll': {

'sprinkle': False,

'advice\_en': 'Healthy cotton boll in development stage. No treatment needed.',

'advice\_hi': 'विकास के चरण में स्वस्थ कपास की टिंडी। किसी उपचार की आवश्यकता नहीं है।',

'pesticide\_en': 'None required',

'pesticide\_hi': 'आवश्यकता नहीं',

'severity': 'None'

},

'Healthy': {

'sprinkle': False,

'advice\_en': 'Healthy cotton plant with no signs of disease. Continue good practices.',

'advice\_hi': 'स्वस्थ कपास का पौधा जिसमें रोग के कोई लक्षण नहीं हैं। अच्छी प्रथाएं जारी रखें।',

'pesticide\_en': 'None required',

'pesticide\_hi': 'आवश्यकता नहीं',

'severity': 'None'

},

'Powdery mildew': {

'sprinkle': True,

'advice\_en': 'Fungal disease appearing as white powdery spots on leaves and stems.',

'advice\_hi': 'फफूंदी जनित रोग जो पत्तियों और तनों पर सफेद पाउडर जैसे धब्बे के रूप में दिखाई देता है।',

'pesticide\_en': 'Sulfur, Myclobutanil, or Tebuconazole',

'pesticide\_hi': 'सल्फर, माइक्लोब्यूटानिल, या टेबुकोनाजोल',

'severity': 'Moderate'

},

'Target spot': {

'sprinkle': True,

'advice\_en': 'Fungal disease causing target-like spots with concentric rings on leaves.',

'advice\_hi': 'फफूंदी जनित रोग जो पत्तियों पर निशाने जैसे धब्बे पैदा करता है जिनमें संकेंद्रित वलय होते हैं।',

'pesticide\_en': 'Chlorothalonil, Azoxystrobin, or Pyraclostrobin',

'pesticide\_hi': 'क्लोरोथैलोनिल, एज़ोक्सिस्ट्रोबिन, या पाइराक्लोस्ट्रोबिन',

'severity': 'Moderate'

}

}

# Initialize session state

if 'predictions' not in st.session\_state:

st.session\_state.predictions = []

if 'current\_prediction' not in st.session\_state:

st.session\_state.current\_prediction = None

# API endpoint for receiving predictions

@st.experimental\_memo

def receive\_prediction(prediction\_data):

"""Receive prediction from Raspberry Pi"""

st.session\_state.current\_prediction = prediction\_data

st.session\_state.predictions.append(prediction\_data)

return {"status": "success"}

# Main app

def main():

st.markdown('<h1 class="main-header">🌿 Smart Cotton Doctor</h1>', unsafe\_allow\_html=True)

st.markdown('### Real-time Cotton Disease Detection & Recommendation System')

# Create tabs

tab1, tab2, tab3 = st.tabs(["Live Detection", "History", "Disease Information"])

with tab1:

col1, col2 = st.columns([2, 1])

with col1:

st.markdown('<div class="sub-header">Live Camera Feed</div>', unsafe\_allow\_html=True)

# Placeholder for camera feed

camera\_placeholder = st.empty()

camera\_placeholder.info("Waiting for Raspberry Pi camera connection...")

# Capture button

if st.button("📷 Capture & Analyze", use\_container\_width=True):

# This would trigger the Raspberry Pi to capture an image

st.info("Sending capture command to Raspberry Pi...")

# In a real implementation, you'd send a command to the Pi here

with col2:

st.markdown('<div class="sub-header">Analysis Results</div>', unsafe\_allow\_html=True)

if st.session\_state.current\_prediction:

prediction = st.session\_state.current\_prediction

disease\_data = disease\_info.get(prediction['disease'], {})

if disease\_data.get('sprinkle', False):

st.markdown(f'<div class="disease-card">', unsafe\_allow\_html=True)

st.error("🚨 DISEASE DETECTED")

else:

st.markdown(f'<div class="healthy-card">', unsafe\_allow\_html=True)

st.success("✅ PLANT IS HEALTHY")

st.metric("Detected Disease", prediction['disease'])

st.metric("Confidence Level", f"{prediction['confidence']}%")

if disease\_data.get('sprinkle', False):

st.markdown(f'<p class="sprinkle-yes">🔴 SPRINKLE: YES</p>', unsafe\_allow\_html=True)

else:

st.markdown(f'<p class="sprinkle-no">🟢 SPRINKLE: NO</p>', unsafe\_allow\_html=True)

st.markdown('</div>', unsafe\_allow\_html=True)

# Advice section

st.markdown("### 🌱 Farmer Advice")

col\_advice1, col\_advice2 = st.columns(2)

with col\_advice1:

st.markdown("\*\*English\*\*")

st.info(disease\_data.get('advice\_en', ''))

if disease\_data.get('sprinkle', False):

st.warning(f"\*\*Recommended Pesticide:\*\* {disease\_data.get('pesticide\_en', '')}")

with col\_advice2:

st.markdown("\*\*Hindi\*\*")

st.info(disease\_data.get('advice\_hi', ''))

if disease\_data.get('sprinkle', False):

st.warning(f"\*\*सुझाया गया कीटनाशक:\*\* {disease\_data.get('pesticide\_hi', '')}")

# Severity and timestamp

st.caption(f"Severity: {disease\_data.get('severity', 'Unknown')} • Detected at: {prediction.get('timestamp', '')}")

else:

st.info("No analysis results yet. Capture an image to begin.")

with tab2:

st.markdown('<div class="sub-header">Detection History</div>', unsafe\_allow\_html=True)

if st.session\_state.predictions:

# Convert to DataFrame for display

history\_df = pd.DataFrame(st.session\_state.predictions)

# Add sprinkle recommendation

history\_df['Sprinkle'] = history\_df['disease'].apply(

lambda x: 'Yes' if disease\_info.get(x, {}).get('sprinkle', False) else 'No'

)

# Display table

st.dataframe(

history\_df[['timestamp', 'disease', 'confidence', 'Sprinkle']],

use\_container\_width=True

)

# Download button

csv = history\_df.to\_csv(index=False)

st.download\_button(

label="Download History as CSV",

data=csv,

file\_name="cotton\_disease\_history.csv",

mime="text/csv"

)

else:

st.info("No detection history available yet.")

with tab3:

st.markdown('<div class="sub-header">Cotton Disease Information</div>', unsafe\_allow\_html=True)

selected\_disease = st.selectbox(

"Select a disease to learn more:",

list(disease\_info.keys())

)

if selected\_disease:

info = disease\_info[selected\_disease]

col\_info1, col\_info2 = st.columns(2)

with col\_info1:

st.markdown("\*\*English Information\*\*")

st.write(f"\*\*Disease:\*\* {selected\_disease}")

st.write(f"\*\*Sprinkle Recommended:\*\* {'Yes' if info['sprinkle'] else 'No'}")

st.write(f"\*\*Severity:\*\* {info.get('severity', 'Unknown')}")

st.write("\*\*Advice:\*\*")

st.info(info['advice\_en'])

if info['sprinkle']:

st.write("\*\*Recommended Pesticides:\*\*")

st.warning(info['pesticide\_en'])

with col\_info2:

st.markdown("\*\*Hindi Information\*\*")

st.write(f"\*\*रोग:\*\* {selected\_disease}")

st.write(f"\*\*छिड़काव सिफारिश:\*\* {'हाँ' if info['sprinkle'] else 'नहीं'}")

st.write(f"\*\*गंभीरता:\*\* {info.get('severity', 'अज्ञात')}")

st.write("\*\*सलाह:\*\*")

st.info(info['advice\_hi'])

if info['sprinkle']:

st.write("\*\*सुझाया गया कीटनाशक:\*\*")

st.warning(info['pesticide\_hi'])

# Run the app

if \_\_name\_\_ == "\_\_main\_\_":

main()

```

## 3. FastAPI Server for Communication (`api\_server.py`)

```python

from fastapi import FastAPI, Request

from fastapi.middleware.cors import CORSMiddleware

from pydantic import BaseModel

import json

from typing import Dict, Any

app = FastAPI(title="Cotton Disease Detection API")

# Enable CORS

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"],

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

# Prediction data model

class PredictionData(BaseModel):

disease: str

confidence: float

sprinkle: bool

advice\_en: str

advice\_hi: str

pesticide\_en: str

pesticide\_hi: str

timestamp: str

# Store predictions

predictions = []

@app.post("/api/prediction")

async def receive\_prediction(prediction: PredictionData):

"""Receive prediction from Raspberry Pi"""

predictions.append(prediction.dict())

return {"status": "success", "message": "Prediction received"}

@app.get("/api/predictions")

async def get\_predictions():

"""Get all predictions"""

return {"predictions": predictions}

@app.get("/api/latest")

async def get\_latest\_prediction():

"""Get the latest prediction"""

if predictions:

return predictions[-1]

return {"message": "No predictions available"}

if \_\_name\_\_ == "\_\_main\_\_":

import uvicorn

uvicorn.run(app, host="0.0.0.0", port=8000)

```

## 4. Installation and Setup Script (`setup.sh`)

```bash

#!/bin/bash

# Update system

sudo apt-get update

sudo apt-get upgrade -y

# Install required packages

sudo apt-get install -y python3-pip python3-opencv libatlas-base-dev libjasper-dev libqtgui4 libqt4-test

# Install Python packages

pip3 install tensorflow opencv-python pillow streamlit fastapi uvicorn requests numpy

# Enable camera interface

sudo raspi-config nonint do\_camera 0

# Create project directory

mkdir -p ~/cotton\_disease\_detection

cd ~/cotton\_disease\_detection

# Download model files (replace with your actual model files)

# wget https://your-domain.com/cotton\_disease\_model.h5

# wget https://your-domain.com/class\_names.json

echo "Setup complete! Please place your model files in ~/cotton\_disease\_detection/"

echo "Model files needed: cotton\_disease\_model.h5 and class\_names.json"

```

## 5. Run Script (`run\_system.sh`)

```bash

#!/bin/bash

cd ~/cotton\_disease\_detection

# Start API server in background

python3 api\_server.py &

# Start Streamlit in background

streamlit run streamlit\_app.py --server.port 8502 --server.address 0.0.0.0 &

# Wait a bit for servers to start

sleep 5

# Start camera detection

python3 pi\_camera\_detection.py

echo "System is running!"

echo "Streamlit UI: http://raspberrypi.local:8502"

echo "API Server: http://raspberrypi.local:8000"

```

## 6. Usage Instructions

1. \*\*Setup Raspberry Pi:\*\*

```bash

chmod +x setup.sh

./setup.sh

```

2. \*\*Place your trained model files in the project directory:\*\*

- `cotton\_disease\_model.h5`

- `class\_names.json`

3. \*\*Run the system:\*\*

```bash

chmod +x run\_system.sh

./run\_system.sh

```

4. \*\*Access the UI:\*\*

- Open a web browser on any device connected to the same network

- Navigate to: `http://raspberrypi.local:8502`

5. \*\*Using the system:\*\*

- The Raspberry Pi camera will be active

- Press 'c' to capture an image and analyze it

- Results will automatically appear in the Streamlit UI

- The UI will show "SPRINKLE: YES/NO" based on the disease

- Detailed advice in both English and Hindi will be provided

## Key Features:

1. \*\*Real-time Detection\*\*: Uses Raspberry Pi camera for live capture

2. \*\*Dual Language Support\*\*: Advice in both English and Hindi

3. \*\*Sprinkle Recommendation\*\*: Clear YES/NO recommendation for pesticides

4. \*\*Historical Data\*\*: Keeps track of all detections

5. \*\*Web Interface\*\*: Accessible from any device on the network

6. \*\*Disease Information\*\*: Comprehensive database of cotton diseases

The system will automatically determine whether sprinkling is needed and provide appropriate recommendations based on the detected disease!