Working Code

Plant Disease Detection – CNN Working Code (Example) :-

simple working Python code for project's key feature:- CNN-based plant disease detection. This can serve as part of final submission code.

Program

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

# Parameters

IMG_HEIGHT = 128

IMG_WIDTH = 128

BATCH_SIZE = 32

EPOCHS = 10

# Dataset directories

train_dir = 'dataset/train'

val_dir = 'dataset/val'

# Data preprocessing

train_datagen = ImageDataGenerator(rescale=1./255)

val_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(

    train_dir,
```

```
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
val_gen = val_datagen.flow_from_directory(
    val_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
# Build model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
  MaxPooling2D(2,2),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(train_gen.num_classes, activation='softmax')
])
# Compile
```

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# Train
history = model.fit(train_gen, epochs=EPOCHS, validation_data=val_gen)
# Save model
model.save("smart_agri_cnn_model.h5")
# Plot accuracy
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.legend()
plt.title('Accuracy')
plt.show()
# Plot loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.legend()
plt.title('Loss')
plt.show()
```

Expected Directory Structure

```
dataset/
├── train/
|   ├── healthy/
```

```
|   ├── blight/
|   └── spot/
└── val/
    ├── healthy/
    ├── blight/
    └── spot/
```

Another Simple Working Program on single python program concept that combines all key features of Smart Agriculture Garden Ai Project So you can see on One of the unified Working Code.

Project Concept One Program Structure :- Organize a single program like this,

Program,

```
# Import all necessary libraries
import cv2
import tensorflow as tf
import numpy as np
from datetime import datetime
import random
# Placeholder for hardware libraries (for future integration)
# import RPi.GPIO as GPIO  # Example for Raspberry Pi hardware
# ==============================
# MODULE 1: Plant Disease Detection
# ==============================
```

```
def detect_plant_disease(image_path, model_path):
    model = tf.keras.models.load_model(model_path)
    image = cv2.imread(image_path)
    image_resized = cv2.resize(image, (128, 128)) / 255.0
    image_exp = np.expand_dims(image_resized, axis=0)
    prediction = model.predict(image_exp)
    class_idx = np.argmax(prediction)
    confidence = np.max(prediction)
    classes = ['Healthy', 'Blight', 'Spot']  # Example class names
    return f"Disease: {classes[class_idx]} (Confidence: {confidence:.2f})"
# ==============================
# MODULE 2: Soil Health Check (Simulated)
# ==============================
def analyze_soil():
    # Simulate with random nutrient level
    soil_quality = random.choice(['Good', 'Needs Fertilizer', 'Needs Water'])
    return f"Soil health status: {soil_quality}"
# ==============================
# MODULE 3: Smart Irrigation (Simulated)
# ==============================
def smart_irrigation(weather='Sunny'):
    # Decision logic (can be improved with API)
    if weather == 'Rainy':
        return "Irrigation not needed due to rain prediction."
```

```
    else:
        return "Irrigation activated."
# ==============================
# MODULE 4: Night Protection (Simulated)
# ==============================
def night_protection():
    # Simulate detection
    threat_detected = random.choice([True, False])
    if threat_detected:
        return "Alert: Pest detected! Activating protection system."
    else:
        return "Garden secure. No threats detected."
# ==============================
# MAIN PROGRAM
# ==============================
if __name__ == "__main__":
    print("□ SMART AGRICULTURE GARDEN AI SYSTEM □")
    print(f"System Check at: {datetime.now()}")
    # Plant disease detection
    disease_result = detect_plant_disease("sample_plant.jpg", "smart_agri_cnn_model.h5")
    print(disease_result)
    # Soil health analysis
    soil_result = analyze_soil()
    print(soil_result)
```

```
# Smart irrigation decision
irrigation_result = smart_irrigation(weather='Sunny')  # You can link to live weather API
print(irrigation_result)
# Night-time protection
protection_result = night_protection()
print(protection_result)
```

Thus, are the Working model Program Code for my Hackathon Prohect on the Smart Agriculture Garden Using, the AI-Driven Sustainable and Protected Farming System.