Phase 3: Implementation of Project

Title: AI DRIVEN QUALITY CONTROL IN MANUFACTURING

Objective

The goal of Phase 3 is to implement the core components of the AI-Driven Quality Control System in Manufacturing based on the plans and innovations developed during Phase 2. This includes the development of the AI model for defect detection, deployment of real-time visual inspection, integration with existing manufacturing systems, and implementation of basic data security measures.

1. AI Model Development

Overview

The primary feature of the AI-Driven Quality Control System is its ability to identify manufacturing defects in real-time using image data. In Phase 3, the AI model will be trained and deployed to detect common defects based on predefined quality standards.

Implementation

- **Computer Vision Model**: The system uses deep learning-based computer vision models (e.g., CNNs) to analyze images from the production line and detect surface defects, misalignments, or anomalies.
- **Data Source:** The model is trained using a dataset of annotated images of both defective and defect-free products. Data augmentation will be used to improve generalization**.**

Outcome

By the end of this phase, the AI model will be capable of detecting basic defects with high accuracy, reducing the need for manual inspection and improving overall quality assurance.

2. Real-Time Inspection Interface

Overview

The AI model will be integrated into a real-time inspection interface that enables on-the-fly analysis of products as they move through the manufacturing process.

Implementation

- **Live Feed Analysis:** The system connects to high-resolution cameras placed along the production line. These cameras feed real-time images to the AI model for analysis.
- **Alert System:** The interface will notify operators of detected defects instantly and log incidents for further review.

Outcome
At the end of Phase 3, the system will be operational in a live manufacturing environment, providing immediate feedback on product quality and reducing defective output.

3. Manufacturing System Integration (Optional)

Overview
Integration with existing manufacturing systems (e.g., MES, ERP) is optional in this phase, but groundwork will be laid for full integration in future stages.

Implementation

- **Data Exchange Framework:** Develop a middleware or API interface for future integration with production planning and inventory systems.
- **Basic Connectivity:** If possible, establish limited communication between the AI system and control systems to record inspection data.

Outcome
By the end of Phase 3, the AI system will be ready for full integration with manufacturing infrastructure, with initial hooks or APIs in place.

4. Data Security Implementation

Overview
Protecting manufacturing data, especially defect logs and product information, is essential. Initial data security practices will be implemented during this phase.

Implementation

- **Encryption:** Inspection data and image logs will be encrypted at rest and during transmission.
- **Access Control:** Role-based access control will be implemented to restrict access to sensitive quality reports and inspection data.

Outcome
At the end of Phase 3, the system will comply with basic data security practices, protecting production data and quality metrics from unauthorized access.

5. Testing and Feedback Collection

Overview
Initial testing of the AI quality control system will be conducted to validate its performance in detecting defects and its ease of use for manufacturing staff.

Implementation

- **Pilot Line Deployment**: A small section of the manufacturing line will be used for pilot testing the system's detection capabilities.
- **User Feedback:** Operators and quality inspectors will evaluate the usability and accuracy of the system and provide feedback for refinement.

Outcome
Feedback from Phase 3 will be used to enhance the AI model and the user interface in Phase 4, ensuring better integration and usability.

Challenges and Solutions

1. **Model Accuracy**
    - ***Challenge*:** The model may struggle with detecting rare or subtle defects in early stages.
    - ***Solution*:** Ongoing retraining with additional data and incorporating user feedback will enhance detection precision.
2. **System Integration**

- ***Challenge*:** Limited compatibility with legacy manufacturing systems.
- ***Solution***: Build flexible APIs and middleware to support gradual integration.

3. **User Training**
   - ***Challenge*:** Operators may require training to interpret AI results effectively.
   - ***Solution***: Develop simple UI guides and provide hands-on training during testing**.**

Outcomes of Phase 3

By the end of Phase 3, the following milestones should be achieved:

1. **AI Defect Detection Model:** A working model capable of identifying common product defects.
2. **Real-Time Visual Inspection:** A functional interface to process and evaluate products in real time.
3. **System Integration Framework:** Foundational elements for future integration with manufacturing systems.
4. **Data Security:** Basic data protection mechanisms in place to secure inspection and product data.
5. **Initial Testing and Feedback:** Insights from pilot deployments to guide the next phase.

Next Steps for Phase 4

In Phase 4, the focus will shift toward:

1. **Improving Model Accuracy:** Using feedback and new data to enhance detection capabilities.
2. **Full System Integration:** Connecting the AI system to manufacturing and inventory systems for seamless operation.

3. **Scalability and Optimization:** Ensuring the solution can be deployed across multiple lines or facilities with minimal latency or overhead.

PROGRAM

```
import os

import cv2

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential, load_model

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from cryptography.fernet import Fernet


# -------- CONFIG --------

DATASET_PATH = 'dataset/'

TEST_IMAGE = 'test_images/sample.jpg'

MODEL_PATH = 'quality_control_model.h5'

LOG_PATH = 'inspection_log.enc'

KEY_PATH = 'secret.key'


# -------- STEP 1: TRAIN OR LOAD MODEL --------
```

```
def train_model():
    datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)

    train_data = datagen.flow_from_directory(
        DATASET_PATH,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary',
        subset='training'
    )

    val_data = datagen.flow_from_directory(
        DATASET_PATH,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary',
        subset='validation'
    )

    model = Sequential([
        Conv2D(32, (3,3), activation='relu',
input_shape=(64, 64, 3)),
```

```
        MaxPooling2D(2,2),
        Conv2D(64, (3,3), activation='relu'),
        MaxPooling2D(2,2),
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
    model.fit(train_data, validation_data=val_data,
epochs=5)
    model.save(MODEL_PATH)
    print("[INFO] Model trained and saved.")

def load_or_train_model():
    if os.path.exists(MODEL_PATH):
        print("[INFO] Loading existing model...")
        return load_model(MODEL_PATH)
    else:
        print("[INFO] Training new model...")
        train_model()
        return load_model(MODEL_PATH)
```

```
# -------- STEP 2: INSPECTION --------
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    if img is None:
        raise ValueError("Image not found.")
    img = cv2.resize(img, (64, 64)) / 255.0
    return np.expand_dims(img, axis=0)

def inspect_product(model, image_path):
    img = preprocess_image(image_path)
    prediction = model.predict(img)[0][0]
    return "Defective" if prediction > 0.5 else "Non-
defective"

# -------- STEP 3: ENCRYPTED LOGGING --------
def generate_key():
    if not os.path.exists(KEY_PATH):
        key = Fernet.generate_key()
        with open(KEY_PATH, 'wb') as f:
            f.write(key)
    else:
        with open(KEY_PATH, 'rb') as f:
```

```
            key = f.read()
    return Fernet(key)

def save_encrypted_log(message, fernet):
    encrypted = fernet.encrypt(message.encode())
    with open(LOG_PATH, 'wb') as f:
        f.write(encrypted)
    print("[INFO] Inspection log encrypted and saved.")

# -------- STEP 4: SIMULATED API --------
def send_to_middleware(data):
    print(f"[API] Sending data to manufacturing system:
{data}")

# -------- MAIN FUNCTION --------
def main():
    print("[SYSTEM] Starting Quality Control System...")

    # Load model
    model = load_or_train_model()

    # Inspect sample product
    print(f"[INFO] Inspecting {TEST_IMAGE}...")
```

```
    result = inspect_product(model, TEST_IMAGE)
    print("Inspection Result:", result)

    # Encrypt result
    fernet = generate_key()
    save_encrypted_log(f"Product inspection result: {result}", fernet)

    # Simulate API call
    send_to_middleware({
        "product_id": 101,
        "result": result
    })

if __name__ == '__main__':
    main()
```

OUTPUT

[SYSTEM] Starting AI-Driven Quality Control System...

[INFO] Loading existing model...

[INFO] Inspecting test_images/sample.jpg...

1/1 [==============================] - 0s 24ms/step

Inspection Result: Non-defective

[INFO] Inspection log encrypted and saved.

[API] Sending data to manufacturing system: {'product_id': 101, 'result': 'Non-defective'}