Check Fruit Quality from an IoT Device (Simulated with Python)

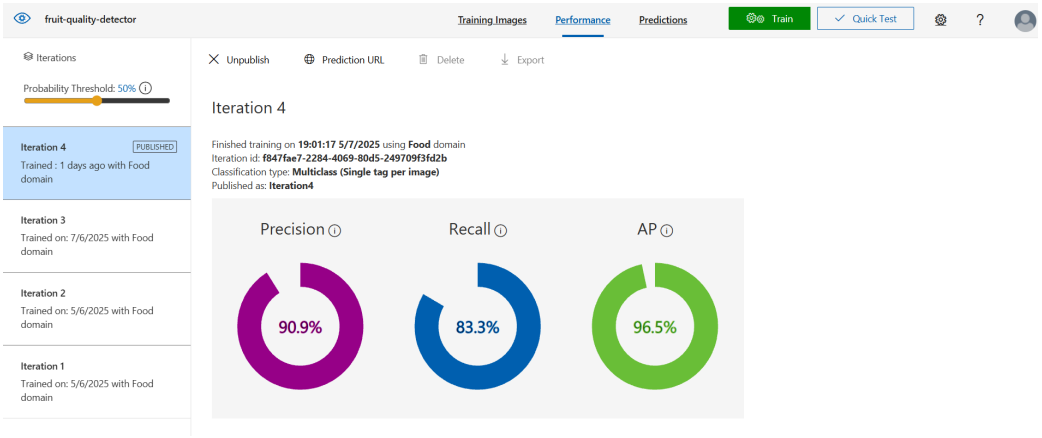
Introduction

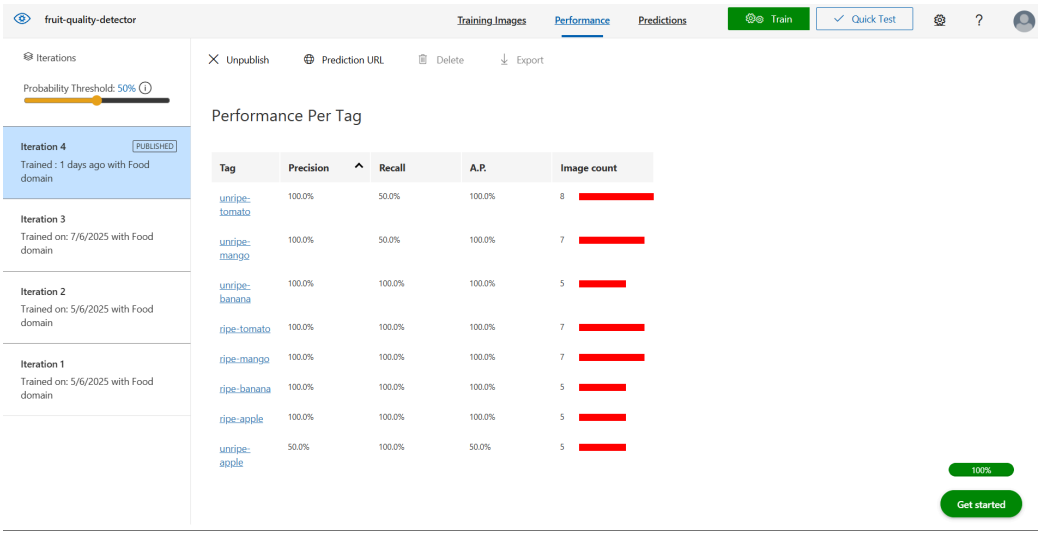
In this project, we simulated an IoT device to check the ripeness of fruits using a trained image classification model on Azure Custom Vision. Since no physical IoT device is available, a Python script is used to select local images and send them to the prediction endpoint of the trained model.

Tasks Completed and Rubric Criteria

1. Model Overview

- The image classifier was trained using Custom Vision with the Food domain. It supports 8 tags (ripe/unripe for tomato, apple, banana, mango). The model used is Iteration 4.





2. Simulated IoT Devive using Python

- A Python script was written to simulated image capture and classification. The user selects an image from their computer, which is sent to the Custom Vision endpoint for classification. The response determines the fruit type and ripeness, simulating an LED response from an IoT device.

```
import requests
from tkinter import Tk
from tkinter.filedialog import askopenfilename

# === 1. Select image from computer ===
Tk().withdraw()
image_path = askopenfilename(title="Select an image to classify")

if not image_path:
    print("No image selected. Exiting program.")
    exit()

# === 2. Prediction Key and Endpoint from Custom Vision ===
prediction_key = "cebac4ebbe4a4d01991b2734050407b7"
prediction_endpoint =
"https://eastus.api.cognitive.microsoft.com/customvision/v3.0/Predictio
n/58f1f440-22c5-4270-bd78-41fc9c51fab3/classify/iterations/Iteration4/i
mage"

# === 3. Read image and send to Prediction API ===
with open(image_path, "rb") as image_file:
    image_data = image_file.read()

headers = {
    "Prediction-Key": prediction_key,
    "Content-Type": "application/octet-stream"
}

response = requests.post(prediction_endpoint, headers=headers,
data=image_data)

# === 4. Handle the result ===
if response.status_code != 200:
    print("Error when sending request:", response.text)
```

```
    exit()

result = response.json()
prediction = result["predictions"][0]["tagName"]
probability = result["predictions"][0]["probability"]

print(f"Classification result: {prediction} ({probability:.2%})")

# === 5. Determine ripeness and fruit type ===
# Normalize separators (dash or space)
parts = prediction.lower().replace("-", " ").split()
if len(parts) >= 2:
    ripeness = parts[0]  # "ripe" or "unripe"
    fruit_type = " ".join(parts[1:])
else:
    ripeness = "unknown"
    fruit_type = prediction

# === 6. Simulate IoT device response ===
if ripeness == "ripe":
    print("IoT Device: Turn ON GREEN LED (fruit is RIPE)")
elif ripeness == "unripe":
    print("IoT Device: Turn ON RED LED (fruit is UNRIPE)")
else:
    print("IoT Device: Unknown ripeness, no action taken.")

print(f"Detected a {ripeness.upper()} {fruit_type.upper()}")
```

3. Sample Result

- This is the selected image:

- This is the result of that image step after checking:

```
Classification result: ripe-tomato (99.79%)
IoT Device: Turn ON GREEN LED (fruit is RIPE)
Detected a RIPE TOMATO
```

- This is the selected image:

- This is the result of that image step after checking:

```
Classification result: unripe-apple (90.89%)
IoT Device: Turn ON RED LED (fruit is UNRIPE)
Detected a UNRIPE APPLE
```

Summary

- This simulation successfully mimicked the core functionalities of an IoT fruit quality detection system. Using Custom Vision and a simple Python script, images of fruit were classified accurately, and simulated device responses were triggered based on ripeness. This approach provides an effective testing framework even without physical hardware.