Plant Disease Detection Using Machine Learning in a Serverless Environment
Configuration Manual

MSc Research Project
Cloud Computing

Peng Yu
Student ID: 22196242

School of Computing
National College of Ireland

Supervisor: Jorge Mario Cortes Mendoza

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Peng Yu
Student ID:	22196242
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Jorge Mario Cortes Mendoza
Submission Due Date:	12/12/24
Project Title:	Plant Disease Detection Using Machine Learning in a Serverless Environment Configuration Manual
Word Count:	405
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Peng Yu
Date:	11th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Plant Disease Detection Using Machine Learning in a Serverless Environment Configuration Manual

Peng Yu
22196242

1 Overview

The Research project is about Using Machine Learning in a Serverless Environment. The training and evaluation of three different Convolutional Neural Networks and the cost and latency comparison between serverless environment (AWS Lambda) and cloud virtual machine (AWS EC2). This configuration Manual has guide to setup the environment and execute the program.

2 System Specifications

2.1 Hardware Requirements

This section provides the details of Hardware and software requirements.

Figure 1 provides the hardware specifications required.

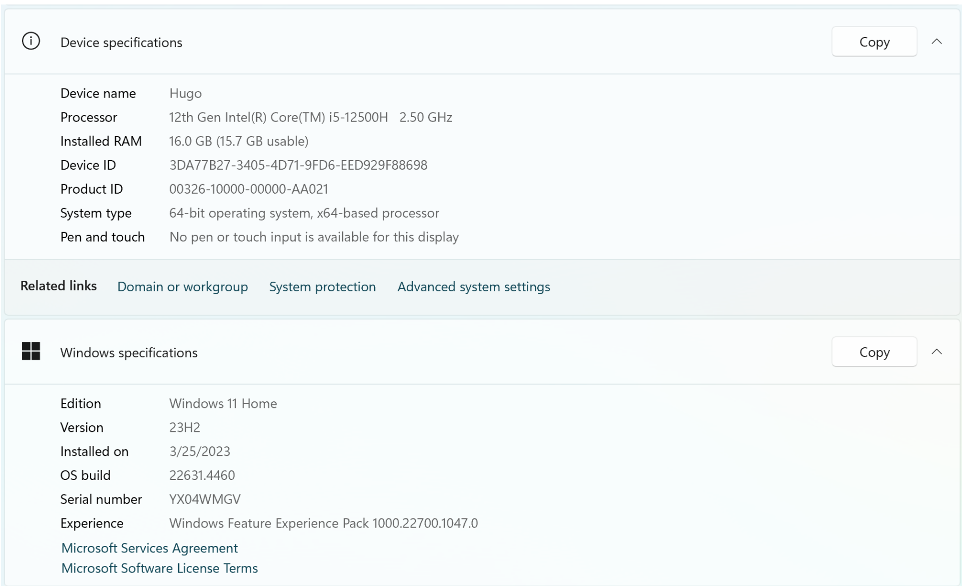

Figure 1: Hardware Requirements

```
=== CPU Information ===
CPU Model: x86_64
Number of Cores: 4

=== GPU Information ===
GPU Available: Yes
GPU Name: Tesla P100-PCIE-16GB

=== Memory Information ===
Total Memory: 33.66 GB
Used Memory: 1.11 GB
Available Memory: 32.07 GB

=== Disk Space ===
Total Disk Space: 8656.92 GB
Used Disk Space: 6456.85 GB
Free Disk Space: 2200.06 GB
```

Figure 2: Kaggle Hardware Requirements

Figure 2 provides the kaggle hardware specifications required.

Additionally, t2.micro EC2 instance and Lambda with 1GB of memory are also required to serve the model.

2.2 Software Requirements

- Model Training
  - Kaggle
- Lambda Deployment
  - onnxruntime
  - Pillow
  - numpy
  - python-jose
  - passlib
- EC2 Deployment
  - fastapi
  - numpy
  - onnxruntime
  - passlib
  - pillow
  - python-jose
  - uvicorn

3 Dataset

The dataset is collected from Github https://github.com/spMohanty/PlantVillage-Dataset

4 Preprocess & Model Training & Model Evaluation

The function has been written and only need to call the function as shown in Figure 3 and modify the model name to execute the entire process seamlessly.

```
data_dir = "/kaggle/input/plantvillage-vages"
model_name = 'mobilenet'  # 可选: 'vgg19', 'mobilenet', 'resnet50'
num_epochs = 40
batch_size = 32
learning_rate = 0.001

print("Starting data preprocessing...")
dataloaders, dataset_sizes, class_names = preprocess_and_split_data(
    data_dir,
    batch_size=batch_size
)

num_classes = len(class_names)
print(f"\nTraining {model_name} for {num_classes} classes...")

# 创建模型和优化器
model = create_model(model_name, num_classes)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# 训练模型
print("\nStarting training...")
model = train_model(
    model,
    dataloaders,
    dataset_sizes,
    criterion,
    optimizer,
    num_epochs=num_epochs
)
```

Figure 3: Preprocess & Model Training & Model Evaluation

5 Model Format Conversion

Use the code in Figure 4 to convert the Pytorch model to ONNX format for deployment.

6 Deployment Configuration

Figure 8 shows the deployment of Lambda function and the ONNX format model. 8 shows the configuration of Lambda function. Figure 7 shows the route configuration of AWS API Gateway. For the EC2 instance deployment, we just need to create a t2.micro

```python
model_name = 'mobilenet'
num_classes = 19

model1 = create_model(model_name, num_classes)
model1.load_state_dict(torch.load('/kaggle/working/mobilenet_plant_disease_detection.pth'))
model1.eval()

import torch.onnx
onnx_model_path = 'mobilenet_plant_disease_detection.onnx'

dummy_input = torch.randn(1, 3, 224, 224)  # (batch_size, channels, height, width)

torch.onnx.export(
    model1,
    dummy_input,
    onnx_model_path,
    export_params=True,
    opset_version=11,
    do_constant_folding=True,
    input_names=['input'],
    output_names=['output']
)
```

Figure 4: Model Format Conversion

EC2 instance and use command ‘uvicorn main:app –host 0.0.0.0 –port 8080 –reload‘ to run the fastapi application.

Figure 5: Lambda Deployment

7 Benchmark: AWS Lambda vs AWS EC2

To compare the latency and cost of AWS Lambda and AWS EC2 for this application, you need to configure the 'ec2_endpoint' and 'lambad_endpoint', then run the python script to performance evaluation. For the token, if you don't configure the Authorization in AWS API Gateway, you don't need to pay attention to the configuration of the token.

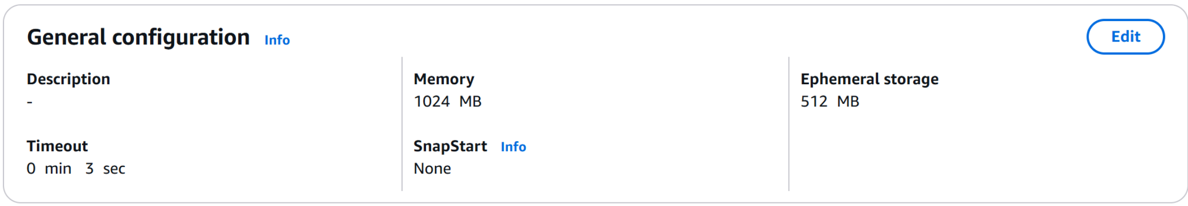

Figure 6: Lambda Configuration

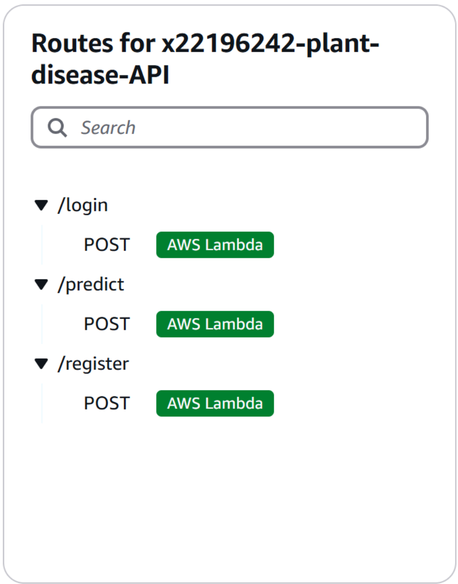

Figure 7: AWS API Gateway

```
class CloudBenchmark:
    def __init__(self, token, test_duration_minutes = 10):
        self.ec2_endpoint = "http://3.254.58.166:8000/predict"
        self.lambda_endpoint = "https://9fkwzv9jm9.execute-api.eu-west-1.amazonaws.com/test/predict"
        self.lambda_headers = {"Authorization": f"Bearer {token}"}
```

```
if __name__ == "__main__":
    token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InVzZXIxIiwiZXhwI
    benchmark = CloudBenchmark(token)

    benchmark.test_duration_minutes = 5

    print("Starting seasonal benchmark...")
    results = benchmark.run_seasonal_benchmark()

    print("\nAnnual Costs:")
    print(f"EC2: ${results['annual_costs']['ec2']['total_cost']:.2f}")
    print(f"Lambda: ${results['annual_costs']['lambda']['total_cost']:.2f}")

    benchmark.plot_results(results)

    timestamp = time.strftime("%Y%m%d-%H%M%S")
    with open(f"benchmark_results_{timestamp}.json", 'w') as f:
        json.dump(results, f, indent=2)
```

Figure 8: Benchmark: AWS Lambda vs AWS EC2