NCHC Open Hackathon

Final Presentation
Plantmen

2024/12/4

Mentor

Jay Chen
&
Cliff Chiu

Members


蔡予恩


林雲


謝宇倫


施宇飛

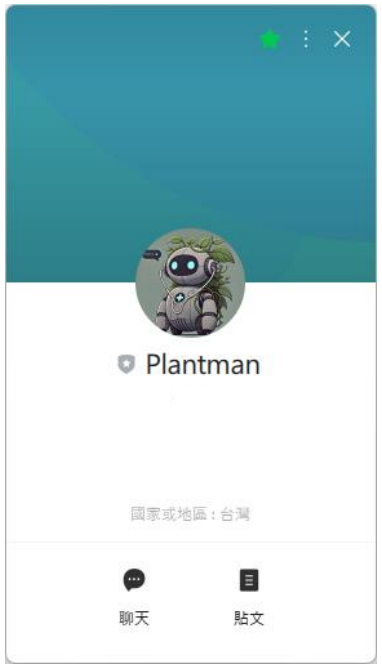

賴聖傑

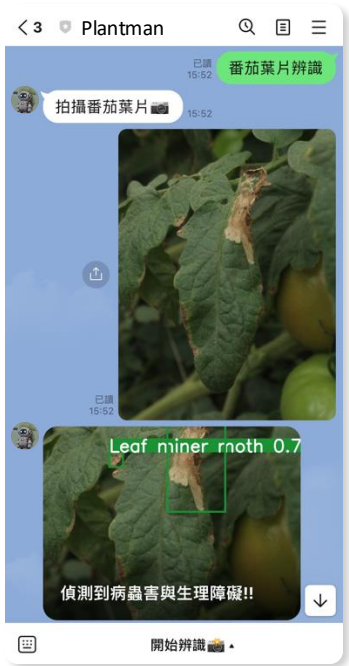
機器學習與機器視覺實驗室
Lab of Machine Learning and Machine Vision

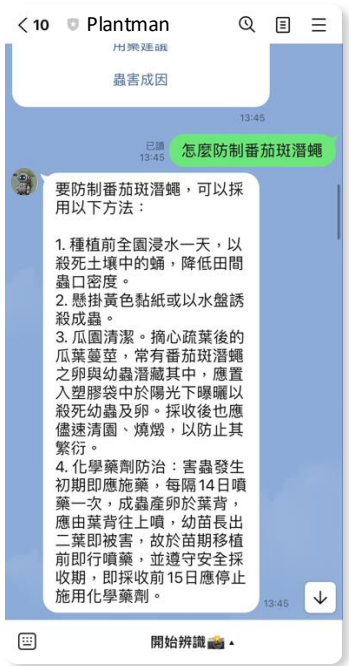
Plantman

Introduction

Plantman is an AI system that combines image recognition and natural language generation to provide **automated disease, pest, and disorder (DPD) identification** and **real-time consultation services** for farmers.





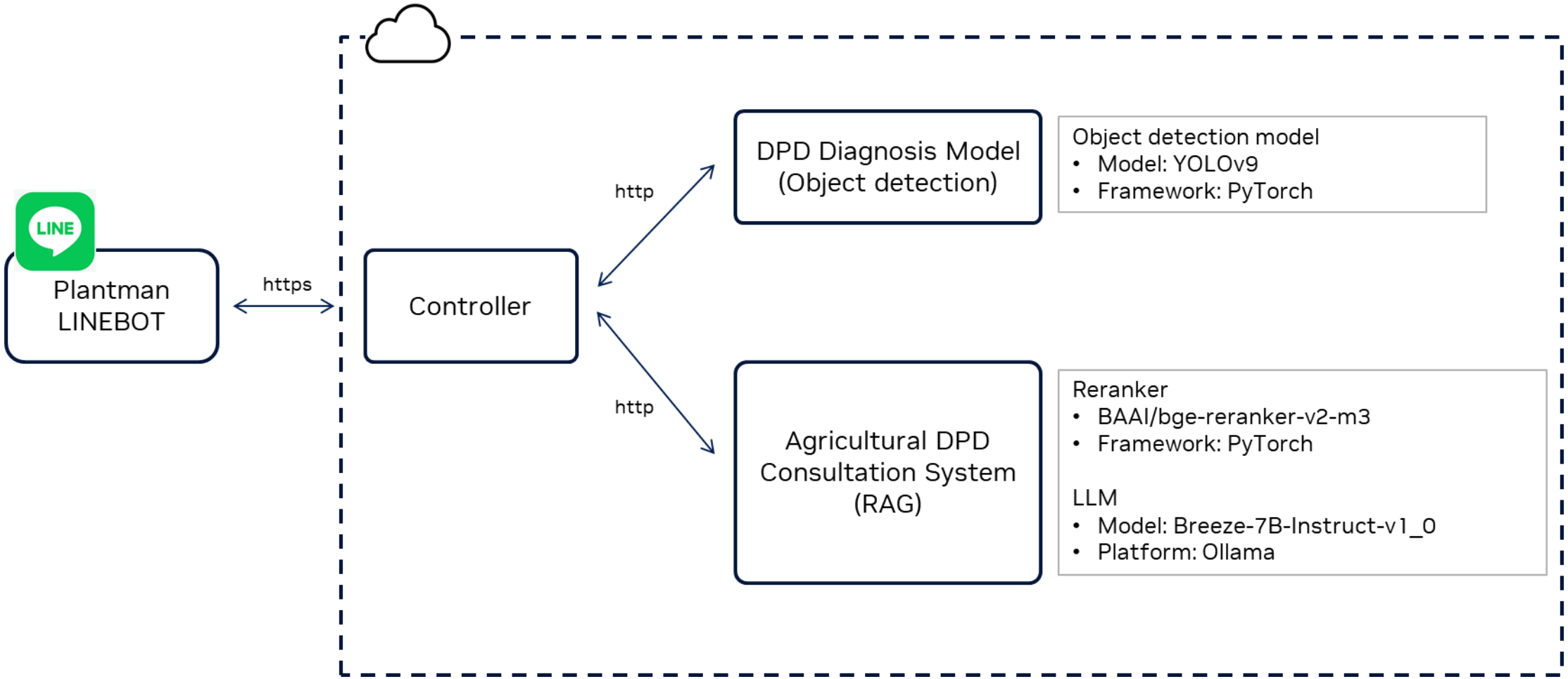


Programming language
Python

Library used
Pytorch, ONNX, CUDA, TensorRT

Algorithmic motifs
Neural Network, Embedding matching, Attention mechanism

Plantman | System architecture



Problem to be solved

1. High user volume causes long query-response delays
2. LINE platform limits streaming, reducing user experience
3. Majority of time consumed by RAG processing

(Speed up 4x)

End-to-end inference time before acceleration

Time Cost of RAG Processing 100 Queries Simultaneously

	RAG	
	Reranker	LLM
Model	BAAI/bge-reranker-v2-m3	Breeze-7B-Instruct-v1_0
Framework	PyTorch (FP32)	Ollama (FP32)
Time cost	1389 sec.	
Speed	13.89 sec./query	

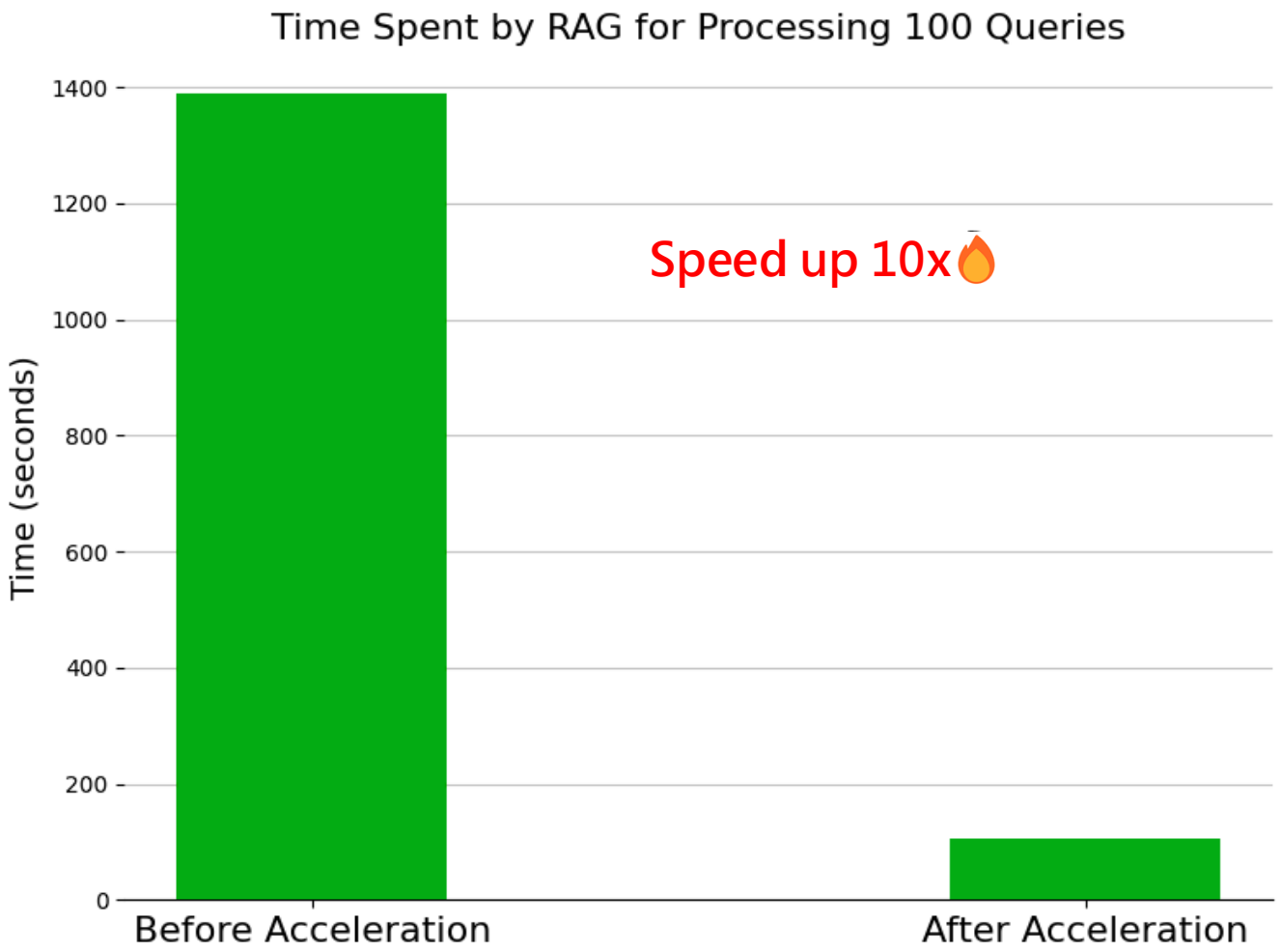
End-to-end inference time after acceleration

Time Cost of RAG Processing 100 Queries Simultaneously

	RAG	
	Reranker	LLM
Model	BAAI/bge-reranker-v2-m3	Breeze-7B-Instruct-v1_0
Framework	TensorRT (FP16)	TensorRT-LLM (INT8)
Time cost	104.47 sec.	
Speed	1.04 sec./query	

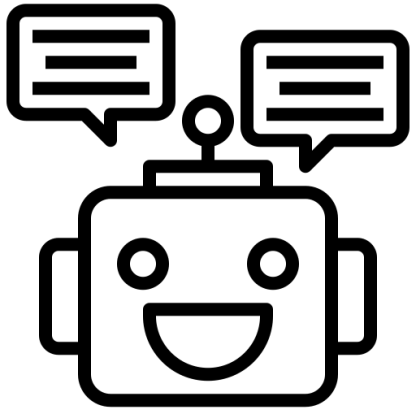
Speed up 10x🔥

End-to-end inference time



Acceleration process

Reranker



LLM

Reranker Acceleration

1. Switch framework from PyTorch to **TensorRT**
2. Deploy inference process on **Triton Server**
3. Performance metric: **sec./query** (time to compare one query with 512 passages)

Time Cost of Reranker Processing 100 Queries Simultaneously

	Throughput (sec./query)	Computing time (sec.)
PyTorch (FP32)	13.78	1378
TensorRT (FP16)	1.17	117
Speedup	11.78x	11.78x

LLM Acceleration

1. Switch platform from Ollama to **TensorRT-LLM**
2. Deploy inference process on **Triton Server**
3. Performance metric: **tokens/sec.**

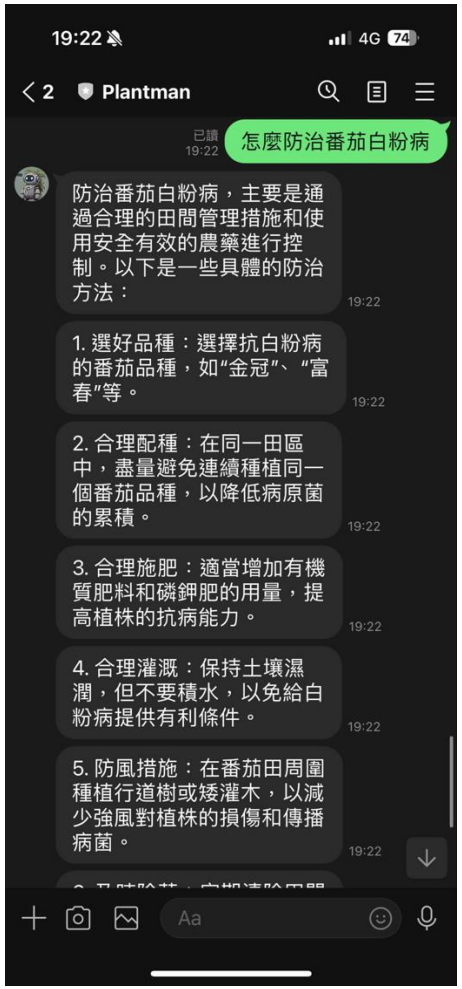
Time Cost of LLM Processing 100 Queries Simultaneously

	Throughput (token/sec.)	Computing time (sec.)
Ollama (FP32)	150.05	880.07
TensorRT-LLM (FP16)	7532.69	17.53
TensorRT-LLM (INT8)	8217.17	15.97
Speedup	55x	55x

LINE user experience optimization

Segment LLM outputs for step-by-step delivery

→ The time it takes for the user to receive the first reply was reduces from **22.94 seconds to 3.98 seconds.**



What do we learn

1. Deployed on Triton Server for efficiency.
2. Converted LLM and reranker weights to TensorRT-LLM and TensorRT .
3. Used Nsight NVTX for performance profiling.

Future Work

1. Migrate entire workflow back to the lab
2. Integrate Embedding model to reduce passage comparisons in RAG

Conclusion

1. Transformed models into TensorRT inference engines.
2. Deployed inference models on Triton Server.
3. Achieved 10x speed-up in E2E inference time.
4. Users now receive replies in just 3 seconds.