

WTMH

An **CWT-CNN** based **AI Server** for **Arrhythmia Screening**
of Real-Time **Single-Lead ECG**

Advisor

Prof. Che-Wei Lin

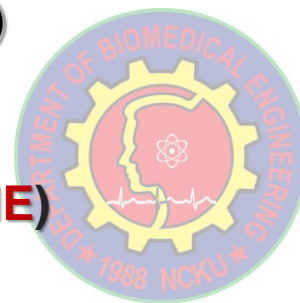
Team Members (Name and organization)

卓子平 朱俊憲 謝侑良
李振揚 黃伊靜 陳政瑋

Department of **Biomedical Engineering (BME)**
National Cheng Kung University (NCKU)

Mentors (Name and organization)

Ken Ying-Kai Liao, NVIDIA



NAR Labs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing



NVIDIA.

Outline

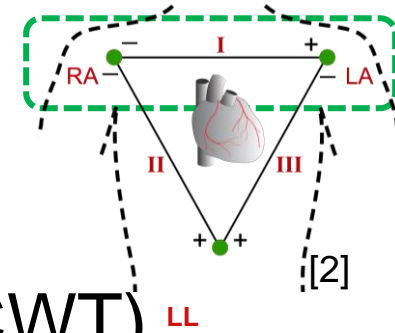
- Introduction
- Architecture Diagram
 - Original Timely System
 - Revised Architecture
- Architecture & Result Comparison
- Issues Faced
- Energy Efficiency
- Conclusion
- Future work

Introduction

- Objective:
 - Develop an **AI server** for real-time **single-lead ECG (單導程心電圖)** analysis for **screening arrhythmia (心律不整)**.



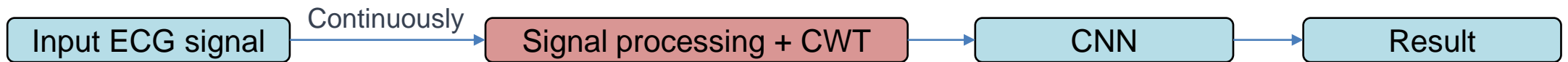
[1]



[2]

- Problem trying to solve

- When performing Continuous Wavelet Transform (CWT) in real-time and large-data detection insufficient processing speed leads to data accumulation.



The CWT is taking up too much time.

Using 3 Hour ECG data to testify: 13.52 sec (segmentation) + **80.52 sec (CWT)** + 8.63 sec (CNN)
(10501 heart beats)

Original Timely System Architecture Diagram

Client

Server

Feature generation: continuous wavelet transform (CWT)

Finding peaks & segmentation

Feature generation

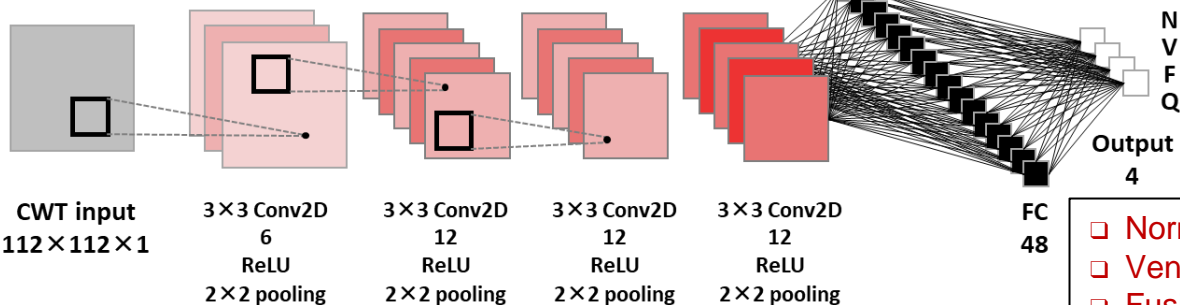
- ECG Signal
- AHA DB

- Windowing
- Normalization
- Low Pass Filter (fc = 20Hz)

- Find R peaks
- Segment to 0.711 sec

- Continuous wavelet transform (CWT)

Classification: SEmbedNet CNN model



- Normal beat
- Ventricular beat
- Fusion beat
- Unknown beat

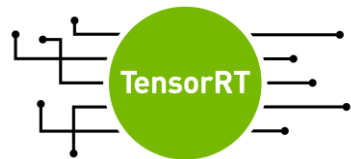
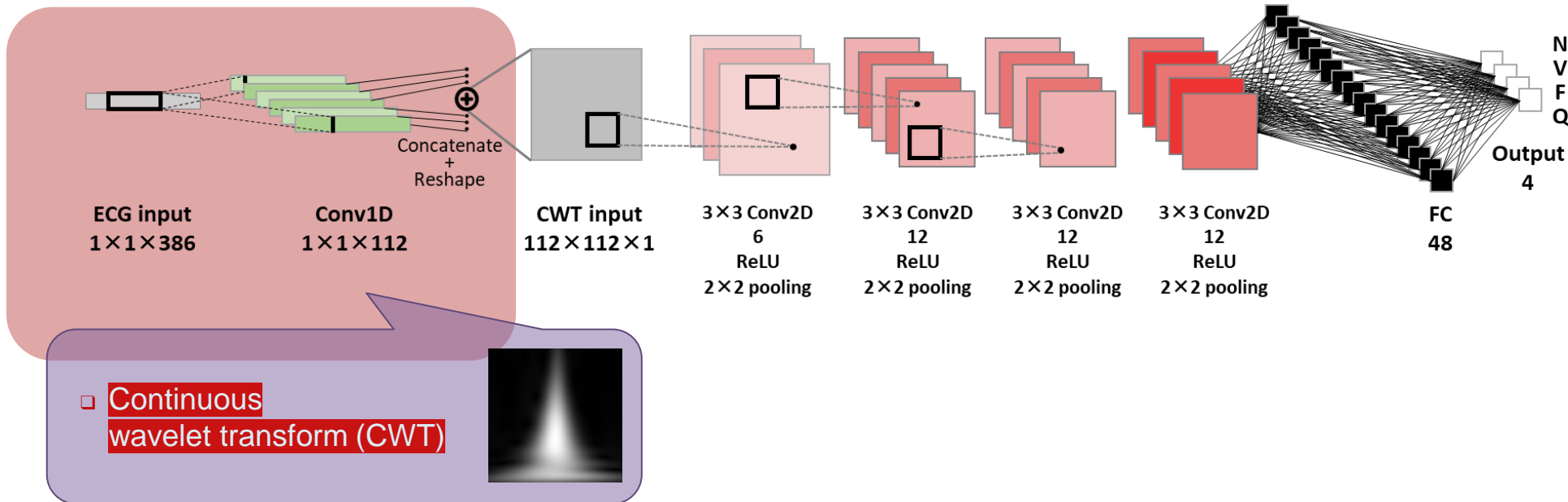
Model	AlexNet	ResNet-50	Proposed CNN
#Parameters	62.4 M	23.9 M	0.04 M

Web UI

Revised Architecture Diagram

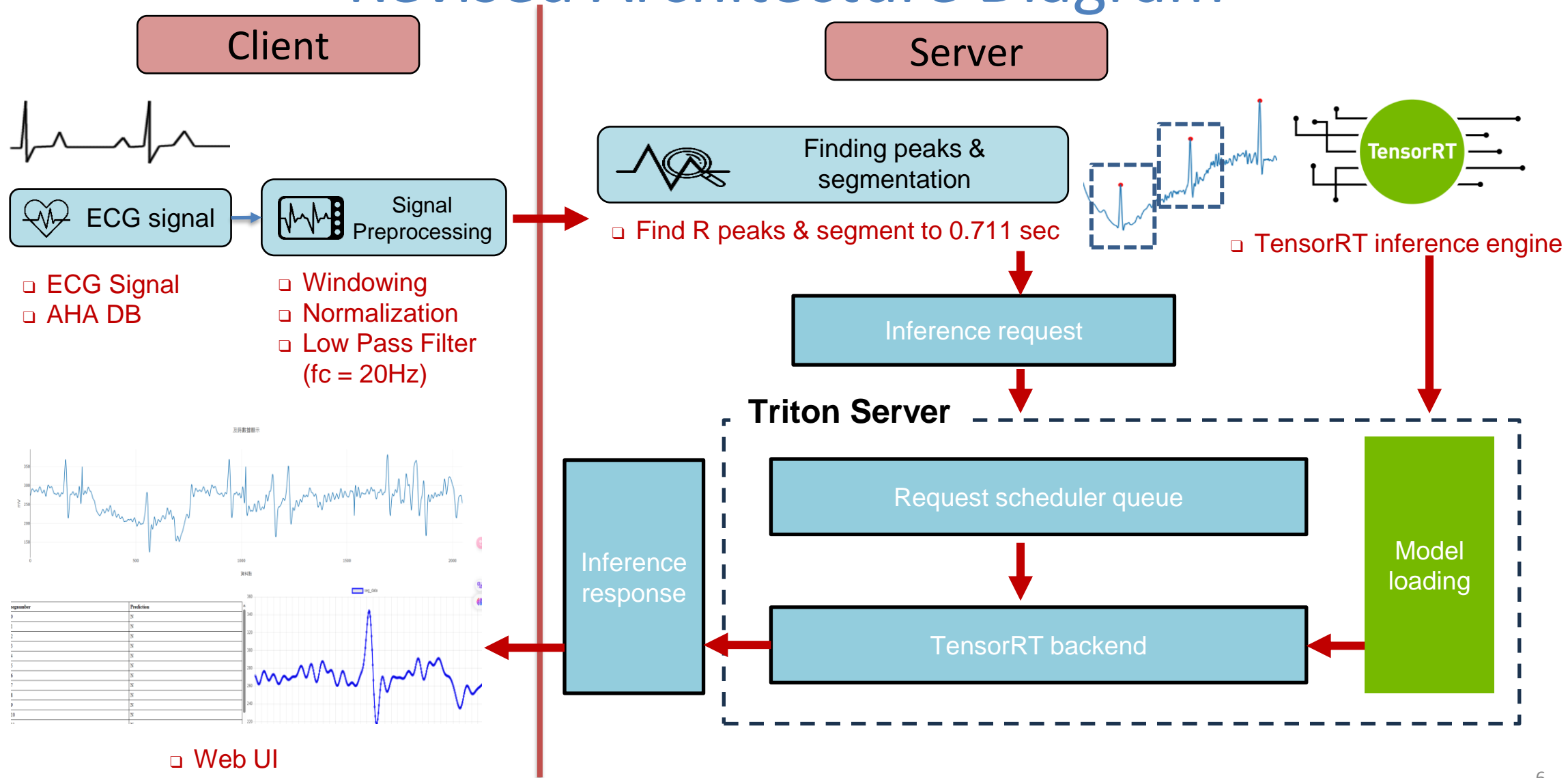
Feature Generation & Classification : combined as a CNN model

**Pytorch Wavelet
acceleration**

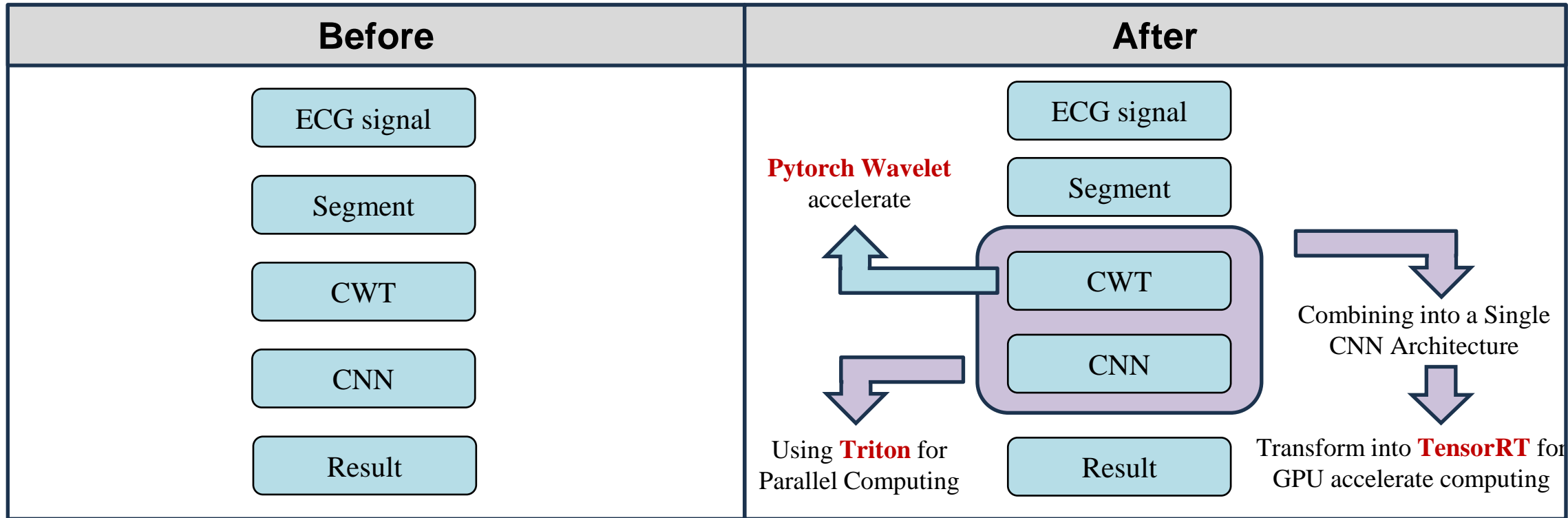


**Transform Pytorch deep learning model into
TensorRT inference engine**

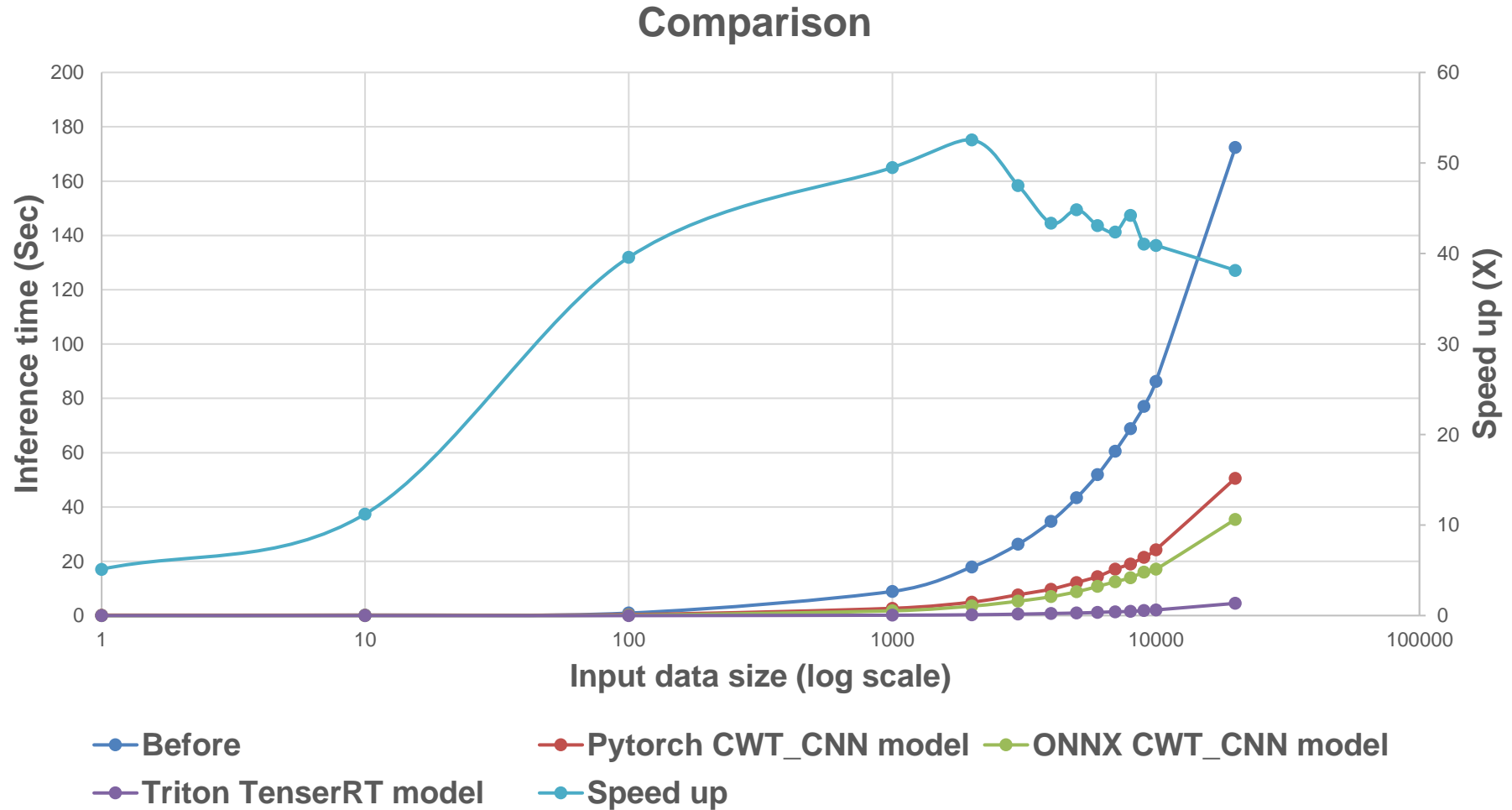
Revised Architecture Diagram



Architecture Comparison Table



Result Comparison Graph



Issues Faced

- The PyTorch-Wavelet package found online do not meet our requirement
 - Selecting a frequency range from 4 to 40
 - Solution: modifying the source code to include frequency range selection
- Unable to compress PyTorch-Wavelet and CNN concatenation into TensorRT
 - Reason: because the parameter of the filter in CWT is not fixed (is constructed when the model is initialized)
 - Solution: extracting the filter parameters into the CWT+CNN model
- We are using float64 as TensorRT input, but the model is built with float32 input, so it cannot get all the input data.

Energy Efficiency

INPUTS	
# CPU Cores	8
# GPUs (A100)	1
Application Speedup	2.8x
Node Replacement	1.4x

GPU NODE POWER SACINGS			
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Saving
Compute Power (W)	1540	6500	-4960
Networking Power (W)	65	93	-25
Total Power (W)	1605	6593	-4988

Node Replacement	0.2x
------------------	------

ANNUAL ENERGY SAVINGS PER GPU NODE			
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Saving
Compute Power (W)	13490	56940	43450
Networking Power (W)	570	814	244
Total Power (W)	14060	57754	43694

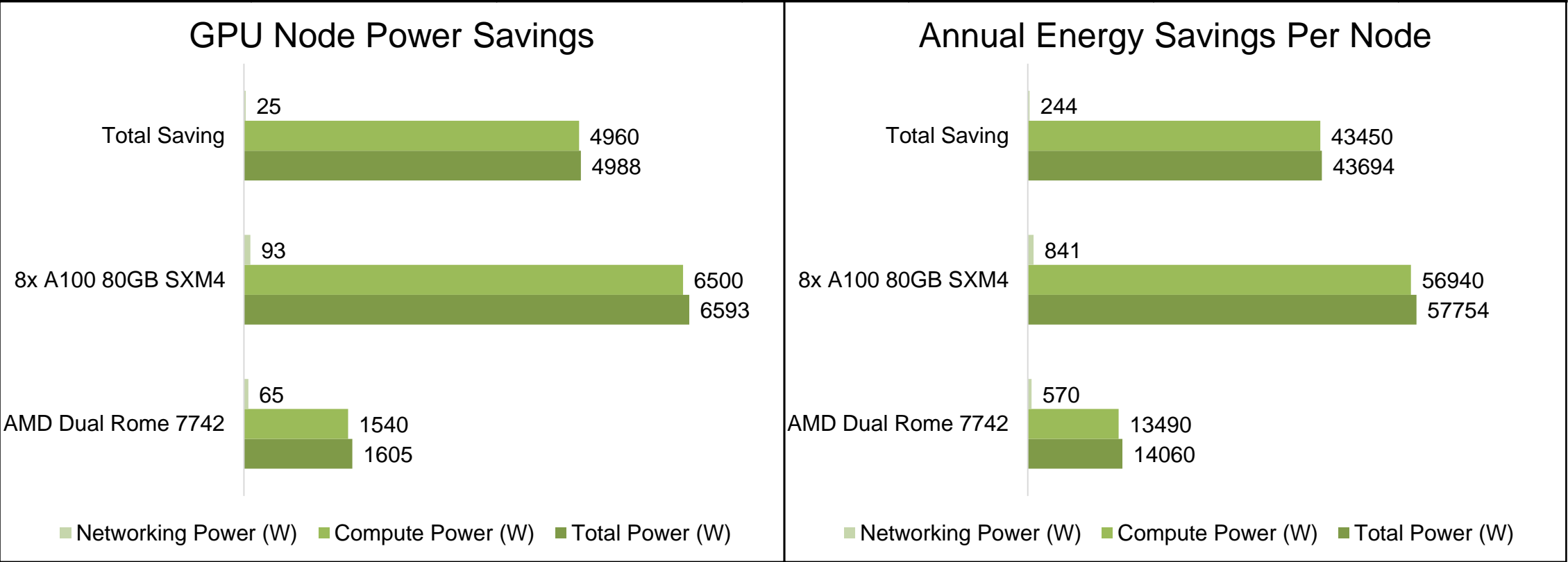
\$/kWh	0.34
Annual Cost Savings	14855.85
3-year Cost Savings	44567.54

Metric Tons of CO ₂	31
Gasoline Cars Driven for 1 year	7
Seedlings Trees grown for 10 years	512

POWER ASSUMPTIONS		
Node Configurations	Baseline Node AMD Dual Rome7742	Alternative 8x A100 80GB SXM4
CPU SKU	7742	7742
# CPU	2	2
# CPU Core	128	128
CPU Power (W)	450	450
GPU SKU	0	A100 80BG SXM4
# GPU	0	8
GPU Power (W)	0	3200
Network Type	IB EDR	IB EDR
#Network Ports	1	2
Network Card Power (W)	30	60
RBoM Power	300	450
Total Compute Node Power (W)	1100	6500
Core Network Power / Node	46	93
Total Power / Node	1146	6593

Energy Efficiency

	GPU NODE POWER SAVINGS			ANNUAL ENERGY SAVINGS PER GPU NODE		
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Saving	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Saving
Compute Power (W)	1540	6500	-4960	13490	56940	43450
Networking Power (W)	65	93	-25	570	814	244
Total Power (W)	1605	6593	-4988	14060	57754	43694



Demo Video



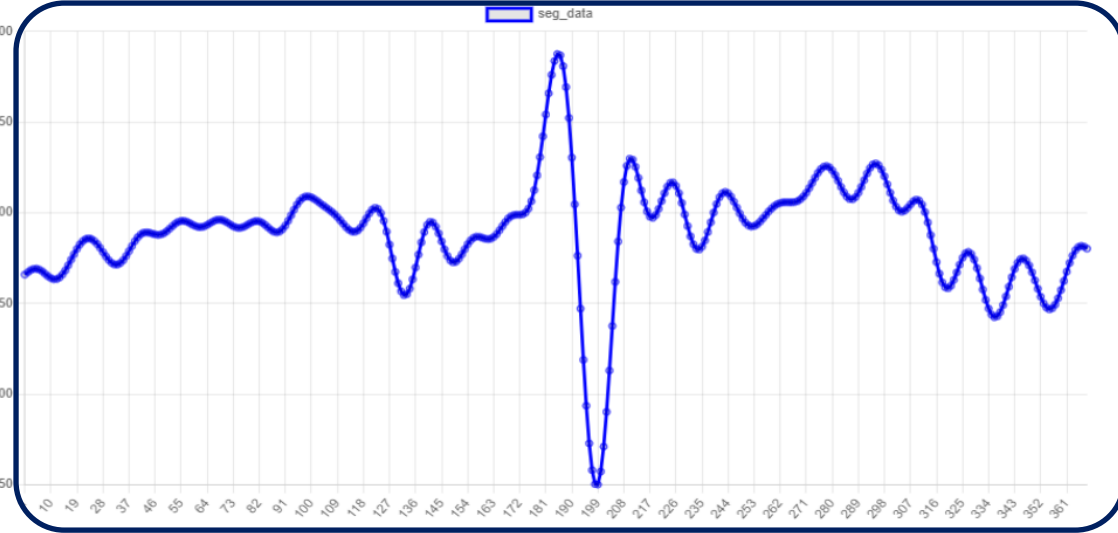
及時數據顯示

ECG signal from real subject



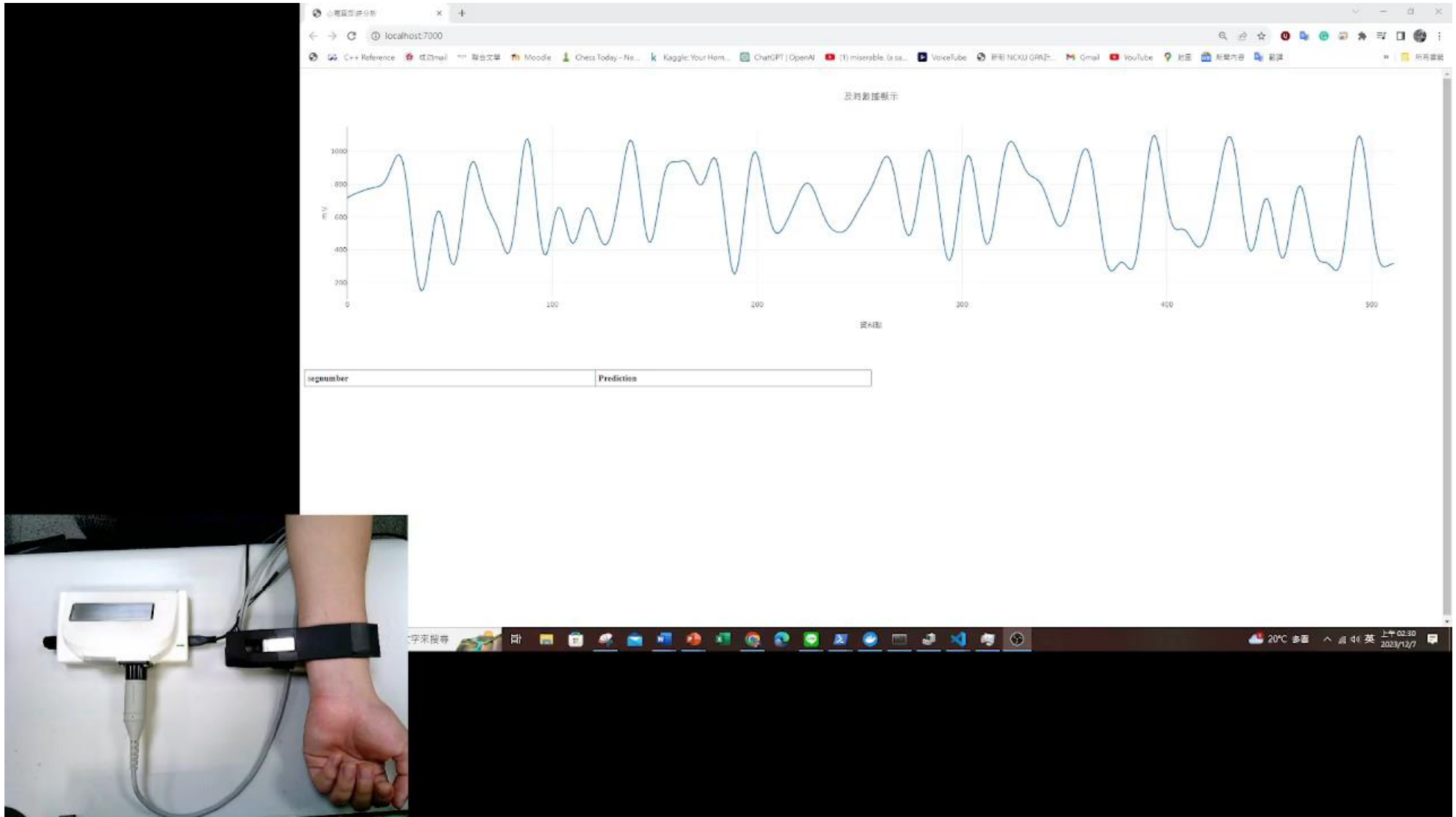
	~
13	N
14	Q
15	Q
16	Q
17	Q
18	Q
19	N
20	N
21	N
22	N
23	N
24	N
25	N

Table of detection results



ECG waveform of selected segment

Demo Video

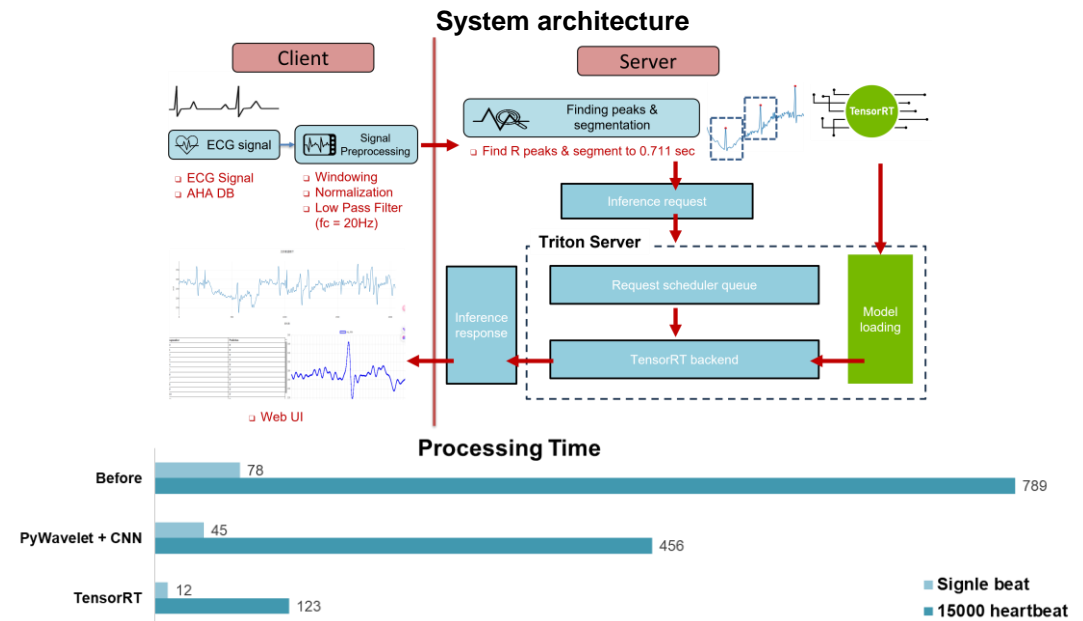


Conclusion

- Conduct CWT transform in Pytorch model format
- Concat CWT and CNN Pytorch model
- Transform the concated model into tensorRT inference engine
- Inference the result through Triton server
- Speed up 40X for Large-ECG-data (3hours up) inference
- Speed up 5x for timely ECG inference

Application Background

- The use of ECG for disease detection is becoming increasingly common.
- If servers are used to provide this service, the following benefits can be realized:
 1. Data storage(SQL)
 2. Retrain model(latest models)
 3. Case Reports
 4. Automatic annotation



Hackathon Objectives and Approach

Objectives

- Accelerate CWT
- More stable system

Approach

- Pytorch-Wavelet
- TensorRT
- Triton inference server

Technical Accomplishments and Impact

- Instant ECG analysis and database creation
- Monitor patient status remotely (Ex: home care & Ambulance ECG connects hospital)
- Patient database
- Convenient long-time ECG data (holter ECG, overnight ECG) annotation

Future Work

- Add more comprehensive CSS and JavaScript
 - Appearance and login interface, etc.
- Enhance cybersecurity-related management
- Enhance the manageability of the server
 - Automate Routine Tasks:
 - Use automation tools to handle repetitive tasks like updates, backups, and monitoring. Scripts and automation platforms can save time and reduce errors.
 - Establish Security Protocols:
 - Implement strict security measures, including firewalls, intrusion detection systems, and regular security audits.
- Extend to 12-leads ECG data

Thanks For Listening

- Code respiratory:
https://github.com/mn12you/wtmh_GUI
- Contact:
NCKU BME WTMH LAB
Prof. Che Wei Lin:
lincw@mail.ncku.edu.tw

