Hybrid Deep Learning and Quantum Network for Typhoon Forecasting in a Wind Power Farm

Team CYCU Power Lab

OpenACC

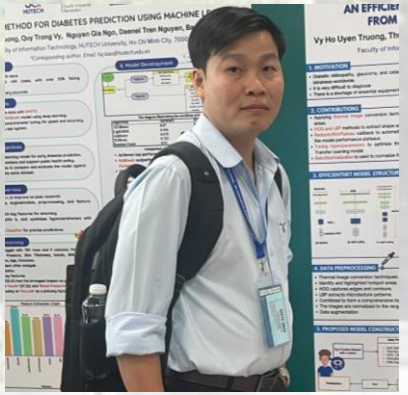
More Science, Less Programming

Team CYCU Power Lab

Yun-Yuan Wang
Mentor from
NVIDIA

Prof. Ying-Yi Hong
Adviser, CYCU

Christian Lian Paulo Perez Rioflorido
Team Leader



Van Huong Phuong
Team Member

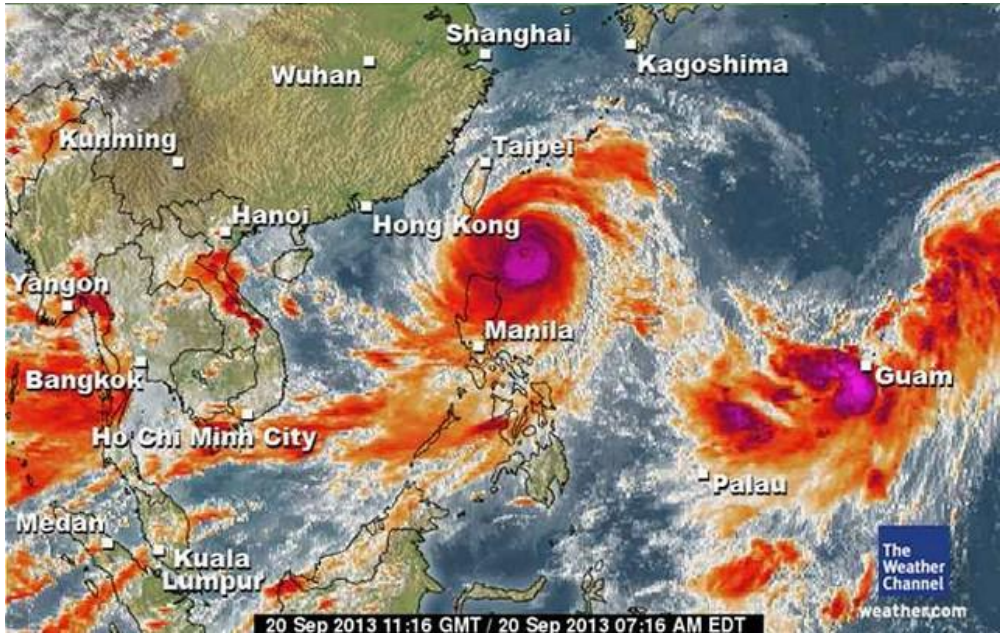
李孟澤
Team Member

中原大學
Chung Yuan Christian University

OpenACC More Science, Less Programming
OPEN HACKATHONS
NCHC 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

Hybrid Deep Learning and Quantum Network for Typhoon Forecasting in a Wind Power Farm

- Our application is a hybrid quantum-classical deep learning system for comprehensive typhoon forecasting and wind farm risk assessment. The system combines **classical deep learning** for multi-horizon attention-based temporal modeling with **quantum neural networks** for quantum-enhanced sequential pattern recognition. We simultaneously forecast typhoon path, radius, intensity, wind speed, and the operational status of wind farm turbines during extreme weather events.



中原大學 Chung Yuan Christian University

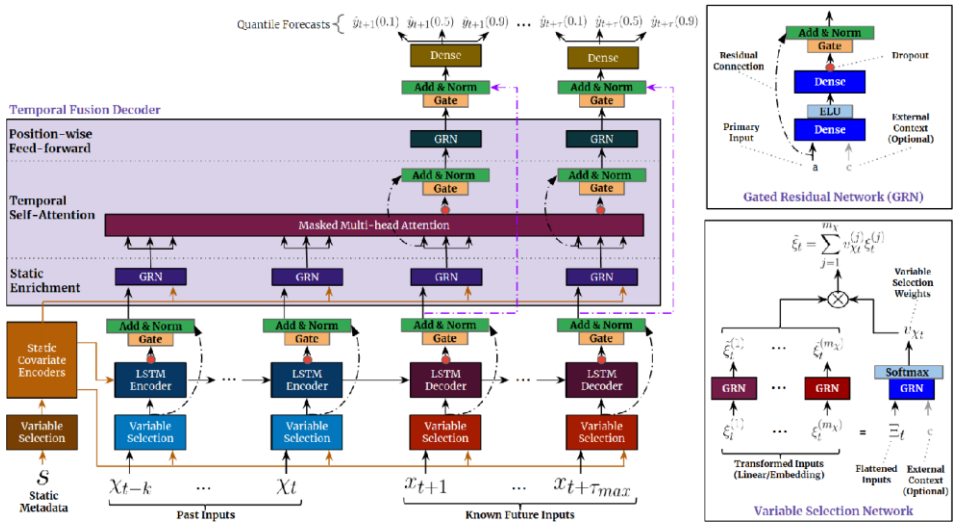
OpenACC More Science, Less Programming | OPEN HACKATHONS | NCHC 國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing | NVIDIA

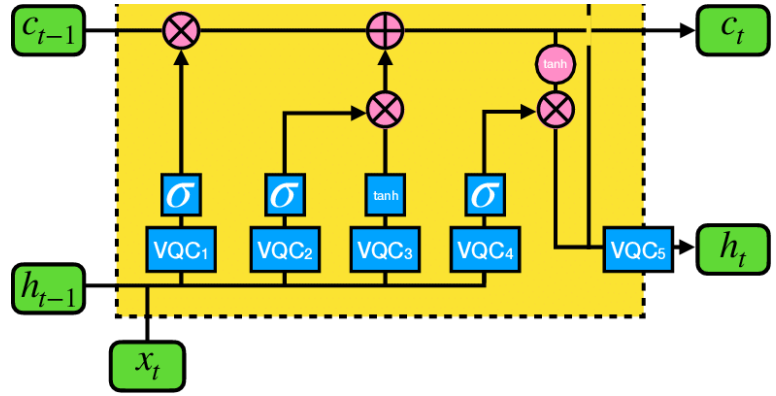
Problem

- Farm-level wind risk from typhoon signals. Inputs: typhoon path (lat/lon), typhoon wind speed, radius (7dir, 10dir); Target: typhoon forecast and wind speed at the wind farm.
- Hackathon goal: maximize training/inference throughput and reduce end-to-end latency across RTX 3080 → A100 → H100, without degrading accuracy.

Scientific Driver

- Temporal Fusion Transformers (TFT) handles multi-horizon temporal dependencies and exogenous selection (great for NWP post-processing).
- Quantum Long Short Term Memory (QLSTM) provides a compact, highly nonlinear temporal inductive bias that can boost tail prediction and calibration, kept fp32 for numeric stability.

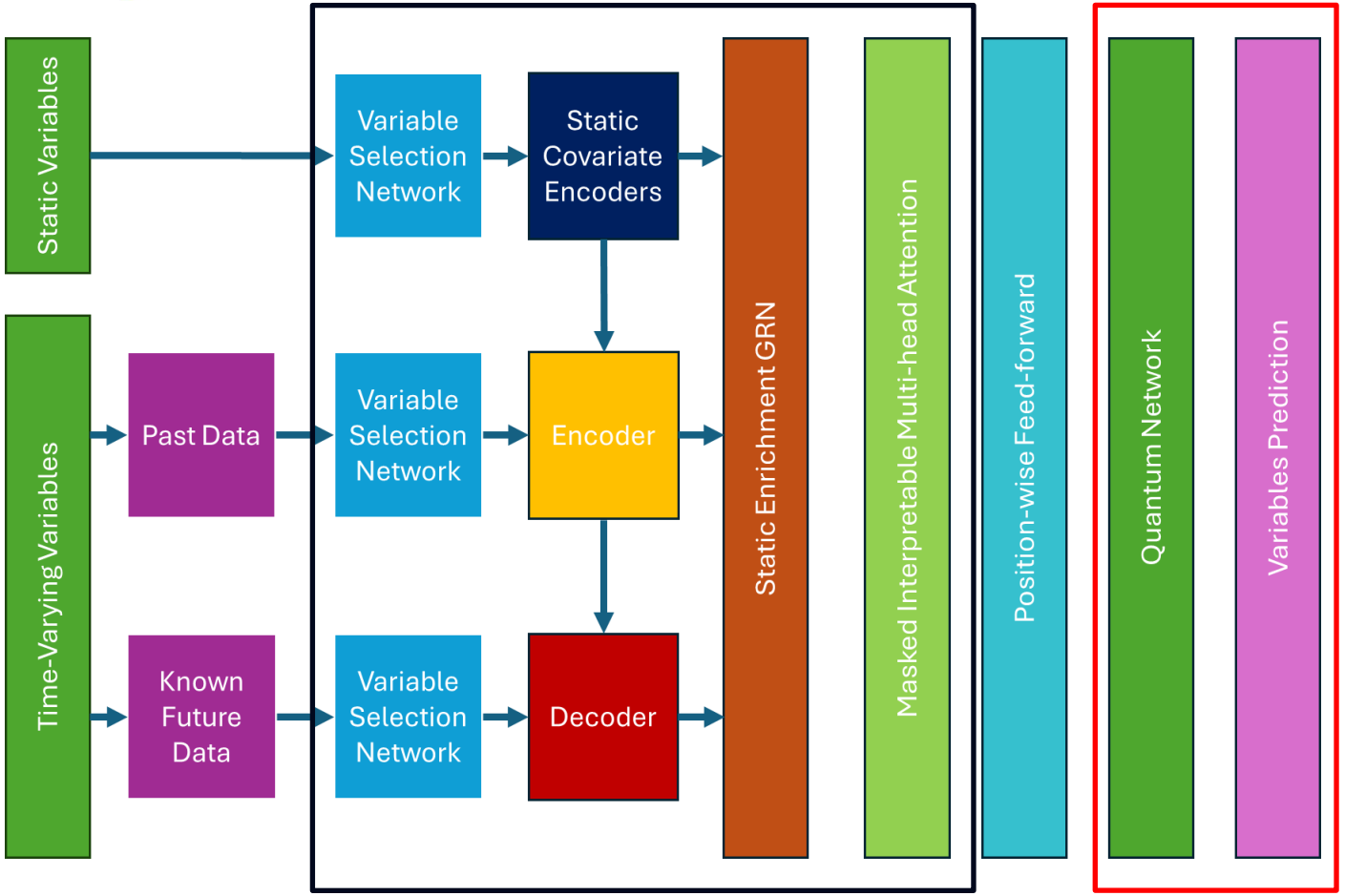




Accelerated by

cuDNN PyTorch

Basic Architecture



- Algorithm Motif: Hybrid Quantum-Classical Multi-Task Architecture:
- Classical TFT Component
- Quantum LSTM

Accelerated by

NVIDIA CUDA QUANTUM

中原大學 Chung Yuan Christian University

OpenACC More Science, Less Programming | OPEN HACKATHONS | NCHC 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

Evolution and Strategy

Goal

- **Turn CWB NWP into farm-scale decisions.** Predict typhoon track (lat/lon), intensity, 7/10-dir radii, and farm wind & gust (multi-horizon).
- **Win on speed**. Demonstrate big end-to-end acceleration across RTX 3080 → A100 → H100 while keeping accuracy within tight guardrails.
- **Make it deployable.** Low-latency inference suitable for operations (curtailment, yaw/park, crew windows).

Evolution and Strategy

Initial strategy

- **Model:** Hybrid **TFT + QLSTM**, plan to sweep **qubits 4–64** and **layers 1–5**, multi-task quantile losses.
- **Scale: DDP on A100/H100**, classical path **bf16-mixed**, quantum path **fp32**; large global batch via grad accumulation.
- **Kernels:** torch.compile (max-autotune), Flash/SDPA attention, fused optimizers.
- **Ablations:** Compare quantum placement (pre/post-attention), encodings, and compute-matched classical baselines.

Evolution and Strategy

How the strategy changed

- **Stability over ambition.** DDP + bf16 in quantum blocks and container issues caused friction (NCCL timeouts, dtype fallbacks), so we locked to single-GPU runs for final scoring.
- **Lean quantum.** Fixed QLSTM to 4 qubits, 1 layer (fp32) for reliability and simpler profiling.
- **Throughput first.** Focused on precision policy per GPU (3080 fp16+TF32; A100/H100 bf16-mixed), Flash-SDPA, torch.compile, and dataloader tuning (pin/persistent/prefetch).
- **Tight targets.** Kept full target set (lat, lon, typhoon wind, 7dir, 10dir, farm wind/gust) but simplified heads/loss masking for missing radii.
- **Metric framing.** Reported epoch-averaged throughput/latency (training-only) as the scoreboard, with inference estimates for bs=1/128.

MAE measures average absolute error, RMSE is like MAE but penalizes large errors more, and R^2 shows variance explained—so smaller MAE/RMSE (closer to 0) and R^2 nearer to 1 mean better performance.

	MAE			RMSE			R2		
	RTX3080	**A100**	**H100**	**RTX3080**	**A100**	**H100**	**RTX3080**	**A100**	**H100**
Longitude	**1.9526e-4**	1.9724e-4	1.9922e-4	**3.2416e-4**	3.7164e-4	4.1912e-4	0.9279	0.9836	**0.9986**
Latitude	**2.2198e-4**	2.2749e-4	2.3299e-4	**3.1986e-4**	4.0799e-4	4.9611e-4	0.9304	0.9944	**0.9984**
Wind Speed	2.0685e-3	2.061e-3	**2.0538e-3**	**5.3174e-3**	5.7867e-3	6.2559e-3	0.9097	0.9772	**0.9972**
7Dir	2.4946e-3	2.4289e-3	**2.3627e-3**	**0.01116**	0.01437	0.01757	0.9166	0.9753	**0.9853**
10Dir	**2.2015e-5**	8.1751e-4	1.613e-3	**0.01043**	0.01398	0.01752	0.9332	0.9509	**0.9909**
Gaomei Wind Speed	**8.1471e-3**	8.217e-3	8.2879e-3	**0.01740**	0.01890	0.02041	0.8375	0.9234	**0.9334**
Gaomei Wind Gust	6.731e-3	6.577e-3	**6.4237e-3**	**0.01442**	0.01576	0.01709	0.8824	0.9340	**0.9540**

3.3 days → **5+ hours** → **4 hours**

RTX3080
(FP16 Precision-TF32, Epochs=50)

A100
(BF16-Mixed Precision, Epochs=50)

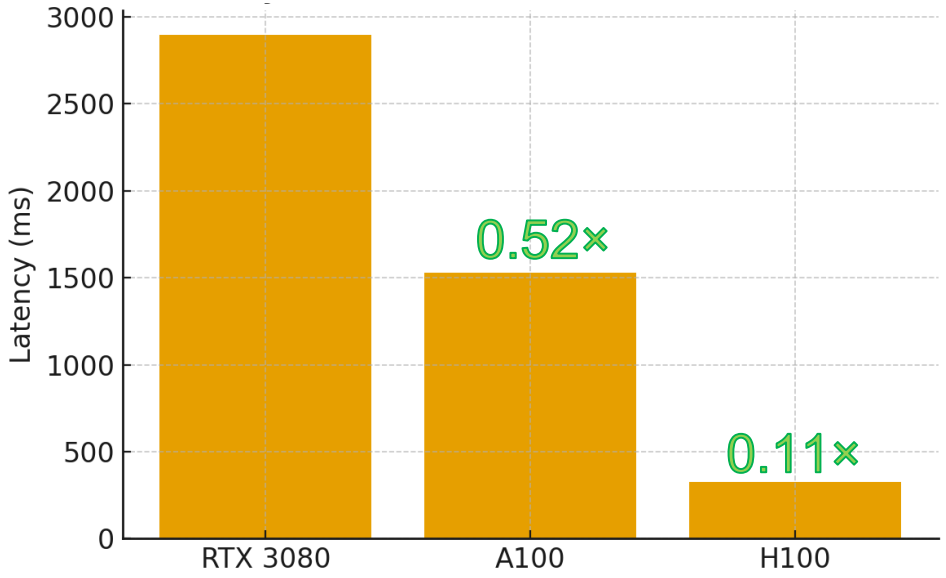
H100
(BF16-Mixed Precision, Epochs=50)

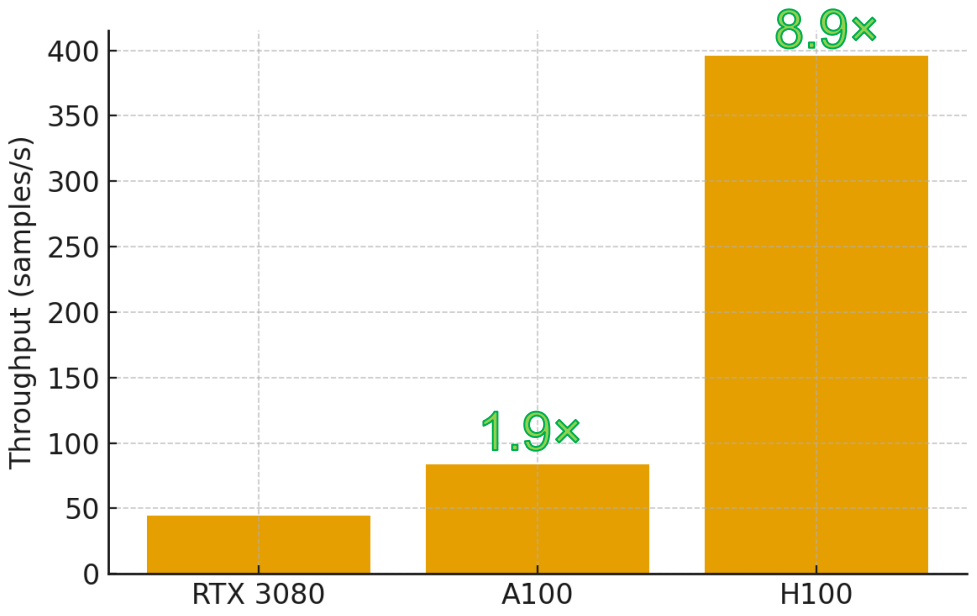
H100 runs reduced computation time and, across most targets, produced lower MAE/RMSE (loss) and R^2 values closer to 1 than the RTX 3080/A100.

中原大學 Chung Yuan Christian University

OpenACC More Science, Less Programming · OPEN HACKATHONS · NCHC 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

Latency and Throughput





Multi-core CPU vs GPU (training, 50 epochs)

Platform	Avg / epoch	50-epoch total	Train it/s (≈)	Speedup vs CPU-16c
CPU (16 cores)	~13.7 h	~28.4 days	~0.008	1.0×
CPU (32 cores)	~9.1 h	~19.0 days	~0.012	1.5×
CPU (64 cores)	~5.5 h	~11.4 days	~0.020	3.4×
RTX 3080	**1h36m**	**3.33 days**	**~0.068**	**11.2×**
A100	**7m**	**5h50m**	**~0.936**	**77.8×**
H100	**5m**	**4h10m**	**~1.31**	**109.2×**

Computation Time Comparison

GPU Model	Configuration	Batch Size	Computation Speed	Status
RTX 3080	4Q / 1L	1024	0.19it/s	✓ Optimal
RTX 3080	Higher configs	—	—	✗ Infeasible
A100	4Q/ 1L	1024	0.20it/s	✓ Baseline
A100	4Q /1L	2048	0.36it/s	✓ Feasible
H100	4Q/ 1L	2048	1.70it/s	✓ Feasible

Energy Efficiency

INPUTS	
# CPU Cores	64
# GPUs (H100)	1
Application Speedup	109.2x
Node Replacement	499.2x

GPU NODE POWER SAVINGS			
	Intel Dual SPR 8480c	**8x H100 80GB SXM4**	**Power Savings**
Compute Power (W)	549,120	9,290	539,830
Networking Power (W)	23,181	186	22,995
Total Power (W)	**572,301**	**9,476**	562,825

Node Power efficiency	**60.4x**

ANNUAL ENERGY SAVINGS PER GPU NODE			
	Intel Dual SPR 8480c	**8x H100 80GB SXM4**	**Power Savings**
Compute Power (kWh/year)	4,810,291	81,380	4,728,911
Networking Power (kWh/year)	203,067	1,627	201,440
Total Power (kWh/year)	**5,013,359**	**83,008**	**4,930,351**

$/kWh	$	0.18
Annual Cost Savings	$	887,463.20
3-year Cost Savings	$	2,662,389.59

Metric Tons of CO_2	3,496
Gasoline Cars Driven for 1 year	754
Seedlings Trees grown for 10 years	57,784

(source: Link)

What problems have you encountered?

- **Legacy app structure**
  - Monolithic script; hard-coded paths/params; .item() in hot path; no feature cache/logging.
  - **Fix:** modularize (data/ model/ train/ utils/), config (Hydra/argparse), remove graph-breaking ops, add cached datasets + unified metrics.
- **Algorithms**
  - QLSTM unstable in bf16; radii (7/10-dir) mask misalign; lat/lon MAE ≠ distance; loss weights drift.
  - **Fix:** keep QLSTM **fp32** (autocast off), strict per-horizon masks, **Haversine** track metric + quantile heads, **uncertainty weighting**, Flash/SDPA + grad-checkpoint.

What problems have you encountered?

- **Tool gaps**
  - CUDA-Q lacks bf16/FP8; no built-in great-circle/fused quantile loss; TFT doesn't expose Flash everywhere.
  - **Fix:** custom Haversine + fused pinball/CRPS, manual SDPA hooks, quantum kept fp32 while classical runs mixed-precision.

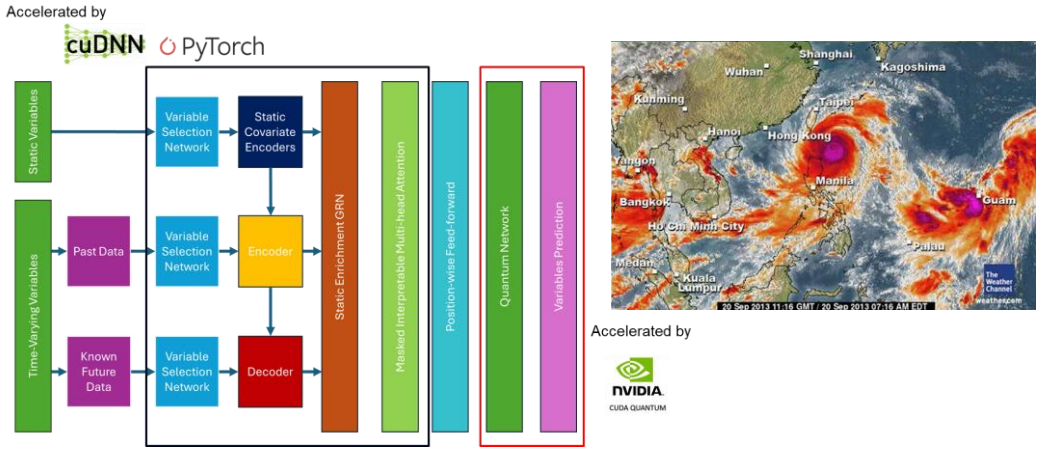
- **System setup**
  - Apptainer missing fuse/binds → no GPU; NCCL timeouts; over-threaded dataloaders → OOM/pickle errors.
  - **Fix:** verify nvidia-smi in container, correct --nv --bind --pwd, install fuse; set NCCL_*, gradient_as_bucket_view=True; workers **8–16/GPU**, pin_memory, persistent_workers, sane prefetch_factor.

- **Tool bugs**
  - "LightningRanger object is not callable"; BF16-unsupported ops in metrics; torch.compile breaks from prints/.item(); Triton first-run hiccups.
  - **Fix:** pin versions, cast metrics to fp32, purge graph-breaking calls, warm-up compile.


Thank you
OpenACC
More Science, Less Programming
中原大學
Chung Yuan Christian University

Hybrid Deep Learning and Quantum Network for Typhoon Forecasting in a Wind Power Farm

A production-oriented forecasting pipeline that predicts typhoon track, arrival time, site-level wind speed/gusts, and risk bands for a wind farm. It blends a proven deep-learning forecaster (for long-range temporal patterns) with a small variational quantum sub-network (for richer, non-linear feature mixing and uncertainty cues). The output supports safer turbine operations, curtailment planning, crew scheduling, grid trading, and early-warning dashboards.



Model → Reality: TFT+QLSTM pipeline (left) forecasting the West Pacific typhoon scene (right) from 72 h history to 24 h ahead.

Hackathon Objectives and Approach

- **Objectives**
- **72→24 hourly forecast** (use last 72 h to predict next 24 h).
- **Accuracy**: lower MAE/RMSE, higher R^2; per-step & arrival-time error.
- **Speed**: maximize **single-GPU** throughput/latency; profile 3080/A100/H100
- **Deliverables**: reproducible pipeline, simple REST/CLI.
- **Approach**
- **Data**: Taiwan **CWB** + best-track; hourly interpolation, cleaning, normalization, route clustering.
- **Model**: **TFT backbone + compact QLSTM** mixer (A/B toggle).
- **Training (single-GPU)**: PyTorch Lightning, **bf16**, Flash/SDPA, grad-checkpointing, torch.compile, **batched quantum** eval using CUDA Quantum
- **Eval/Deploy**: MAE/RMSE/R^2 per variable & step, reliability diagrams; containerized API + lightweight dashboard.

Achieved: Production-style **72→24 h** forecasting pipeline (wind, gust, lat/lon) with TensorBoard, per-step metrics, calibrated arrival time; **better loss and higher R^2** than classical TFT.
How: bf16, SDPA/Flash-Attention, torch.compile, grad-checkpointing, larger stable batches, **batched quantum eval**, Parquet + mem-map, pinned/persistent DataLoaders.
Speedup: 0.19→0.36→1.70 it/s (RTX3080 → A100 → H100) = **1.9×**, **8.9×**, **4.7×**; 50-epoch wall-time ≈ **80 h**, **42.2 h**, **8.9 h**.
Why it matters: Enables **hourly refresh forecasts** with higher confidence for curtailment, crew scheduling, and trading—while **cutting compute time/cost and accelerating experimentation**.

Platform	Avg / epoch	50-epoch total	Train it/s (≈)	Speedup vs CPU-16c
CPU (16 cores)	~13.7 h	~28.4 days	~0.008	1.0×
CPU (32 cores)	~9.1 h	~19.0 days	~0.012	1.5×
CPU (64 cores)	~5.5 h	~11.4 days	~0.020	3.4×
RTX 3080	**1h36m**	**3.33 days**	**~0.068**	**11.2×**
A100	**7m**	**5h50m**	**~0.936**	**77.8×**
H100	**5m**	**4h10m**	**~1.31**	**109.2×**

OpenACC More Science, Less Programming · OPEN HACKATHONS · NCHC 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