Quantum Extreme Learning Machine: Case Studies on Software Testing

4/12/2024 | 15:00 h | Stockholm

Xinyi Wang

Simula Research Laboratory, Norway

University of Oslo, Norway

Quantum Autumn School 2024

Industrial Context

Orona

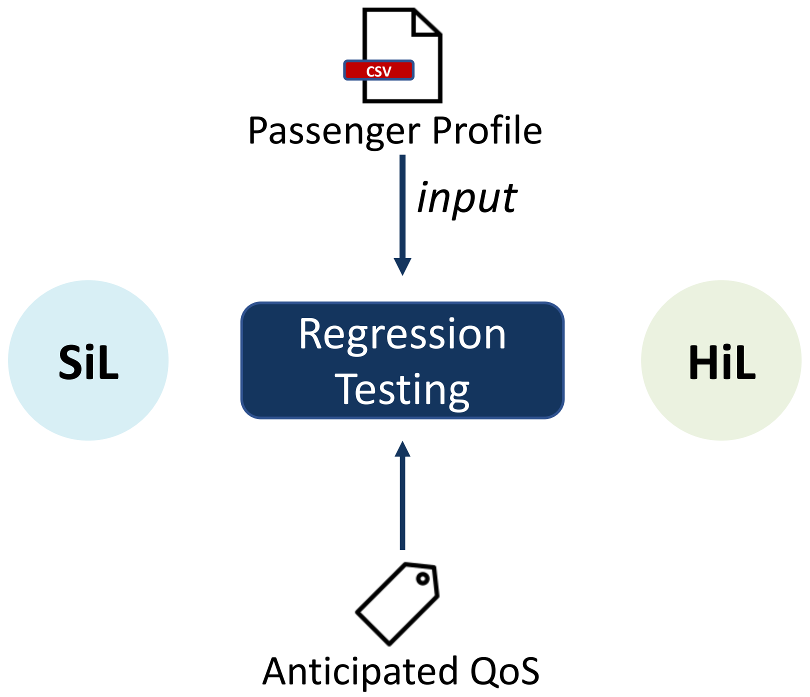
Orona is one of the largest elevator companies in Europe, with over 250,000 installations in the world.

A system of elevators aims to transport passengers as safely as possible while **minimizing the time** they need to wait for the elevator.

Dispatching algorithm is used to schedule the elevators as optimally as possible by assigning an elevator to each call.

Industrial Context

A **dispatching algorithm** undergoes regular maintenance and evolution.



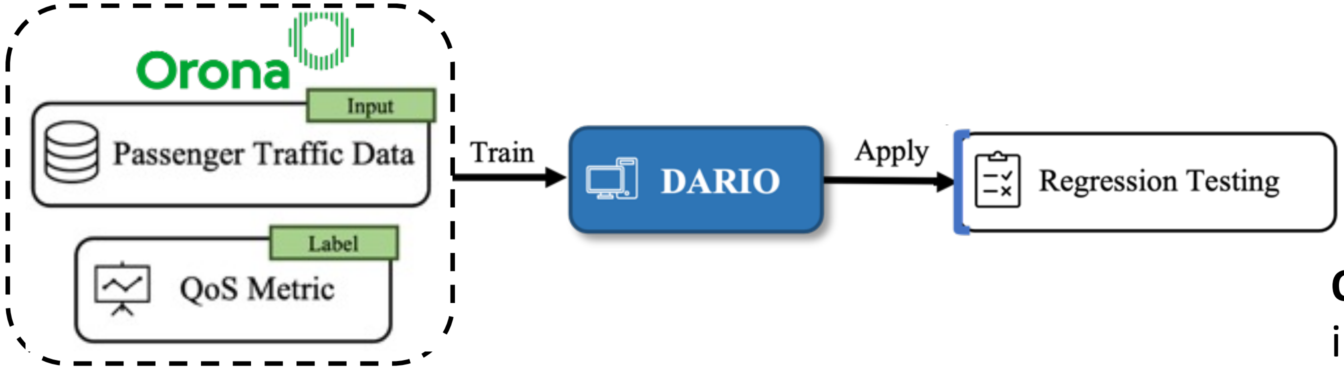
- **Software in the loop (SiL):** a domain-specific simulator, *ELEVATE*[1]
- **Hardware in the loop (HiL):** actual hardware components, e.g., real-time operating systems, and human-machine interace
- **Passenger profile:** passenger information with various attributes, e.g., destination floor, arrival floor, and mass
- **Quality of Service (QoS) metrics:** obtained by re-running the test with a different algorithm or and older version , e.g., average waiting time

- **The cost of re-execution is big.** !!!
- **It's impossible to re-execute any test at operation time.**

Machine learning (ML)-based models are proposed to predict the QoS metrics and replace the regression testing oracle.

[1] Elevate 2024. Elevate. https://peters-research.com/index.php/elevate/

Motivation



Passenger traffic data: extracted from passenger profiles for a time window of 5 minutes.
e.g., number of upward calls, travel distance

QoS Metric: average waiting time (AWT).
i.e., average time passengers wait in a time window of 5 min.

Challenge:

Various installation configurations → Certain **features** of passenger traffic data might be **unavailable**

Operation time → Certain **features** of passenger traffic data might be **unavailable**

!!! Number of features can be used for training is **limited**

One promising solution: Quantum Extreme Learning Machine (QELM)

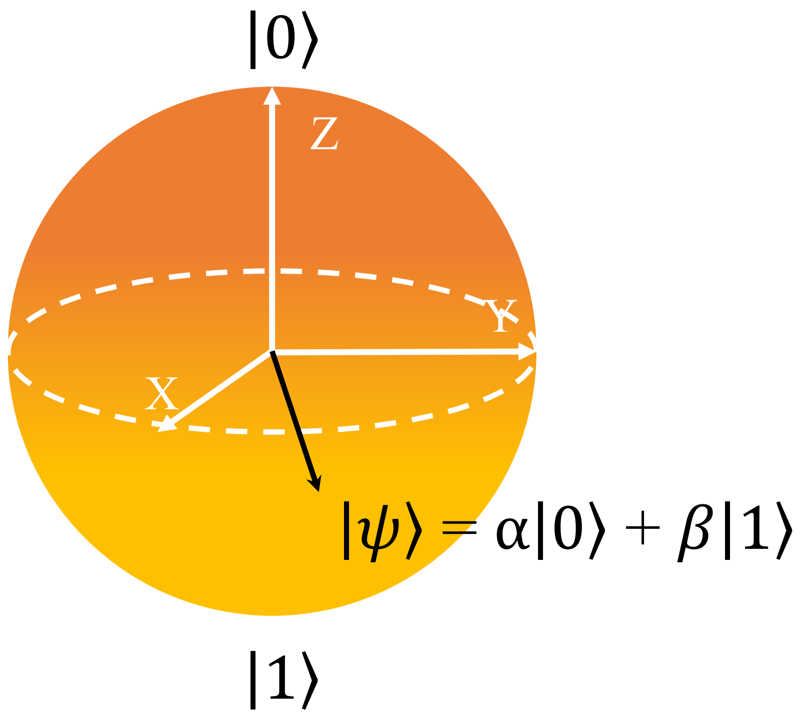
- A quantum machine learning algorithm.
- Maps classical input data into higher-dimensional quantum space using quantum dynamics of **quantum reservoir**.
- Enables efficient **ML** training with **fewer features** while maintaining good prediction quality.

We propose a QELM-based approach, **Q**uantum **E**xtreme **L**earning e**L**evator (QUELL)[2].

[2] Wang, Xinyi, et al. "Application of quantum extreme learning machines for qos prediction of elevators' software in an industrial context." Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering. 2024.

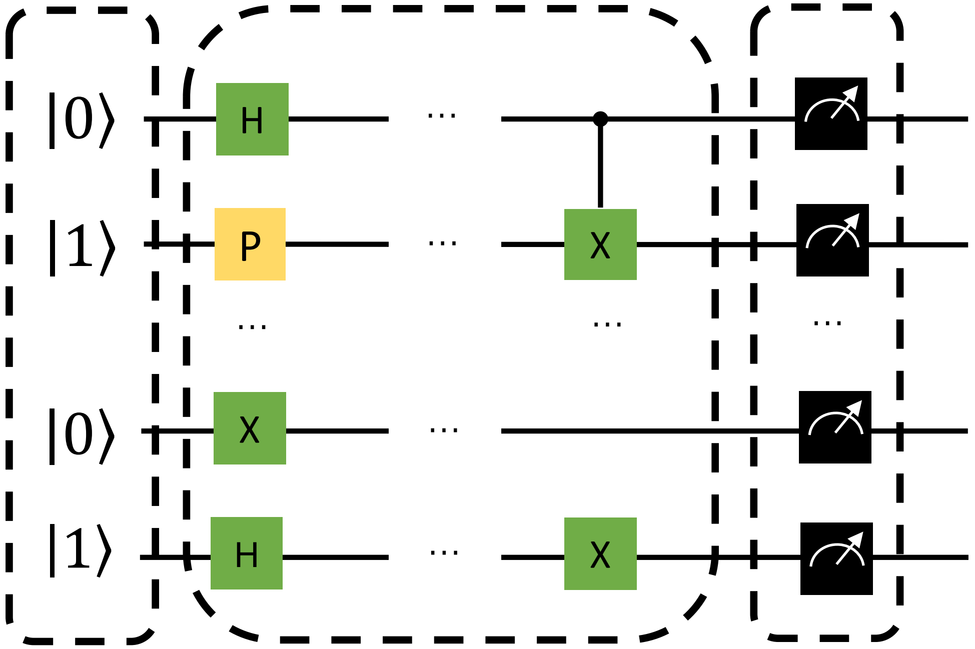
Background – Quantum Computing

Qubit



Superposition

Quantum Circuit



Initialization
(assign 0/1 for each qubit)

Quantum
Operations

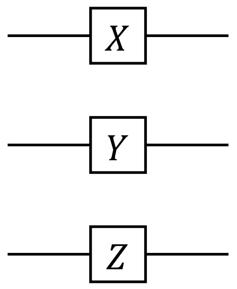
Measurements

Bit

Background - Quantum Gates

Pauli gates:

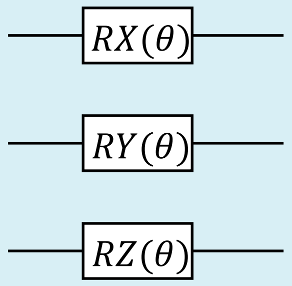
It rotates a qubit around x, y, or z axis with π radians



Encoding

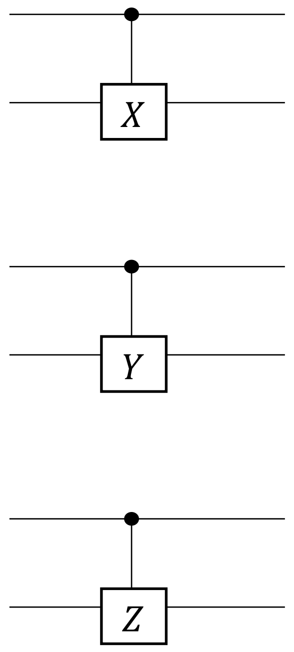
Rotation gates:

It rotates a qubit around x, y, or z axis with θ radians

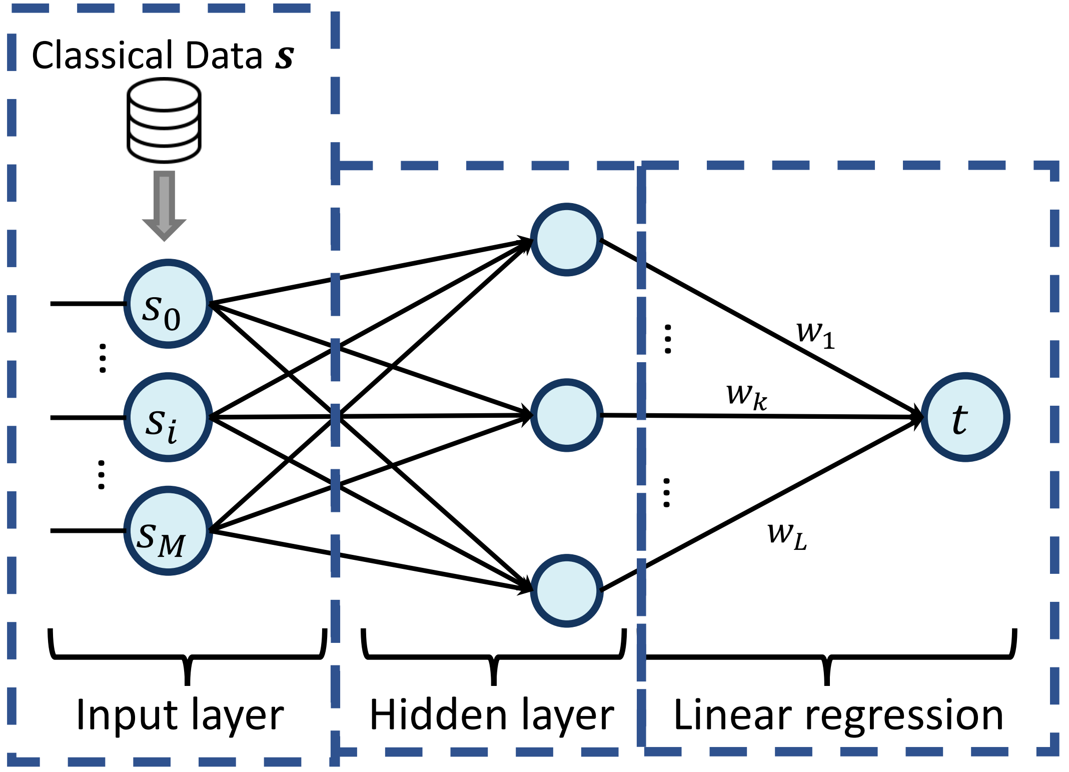


Control gates:

If the control qubit is $|1\rangle$, the target qubit rotates around x, y, or z axis with π radians

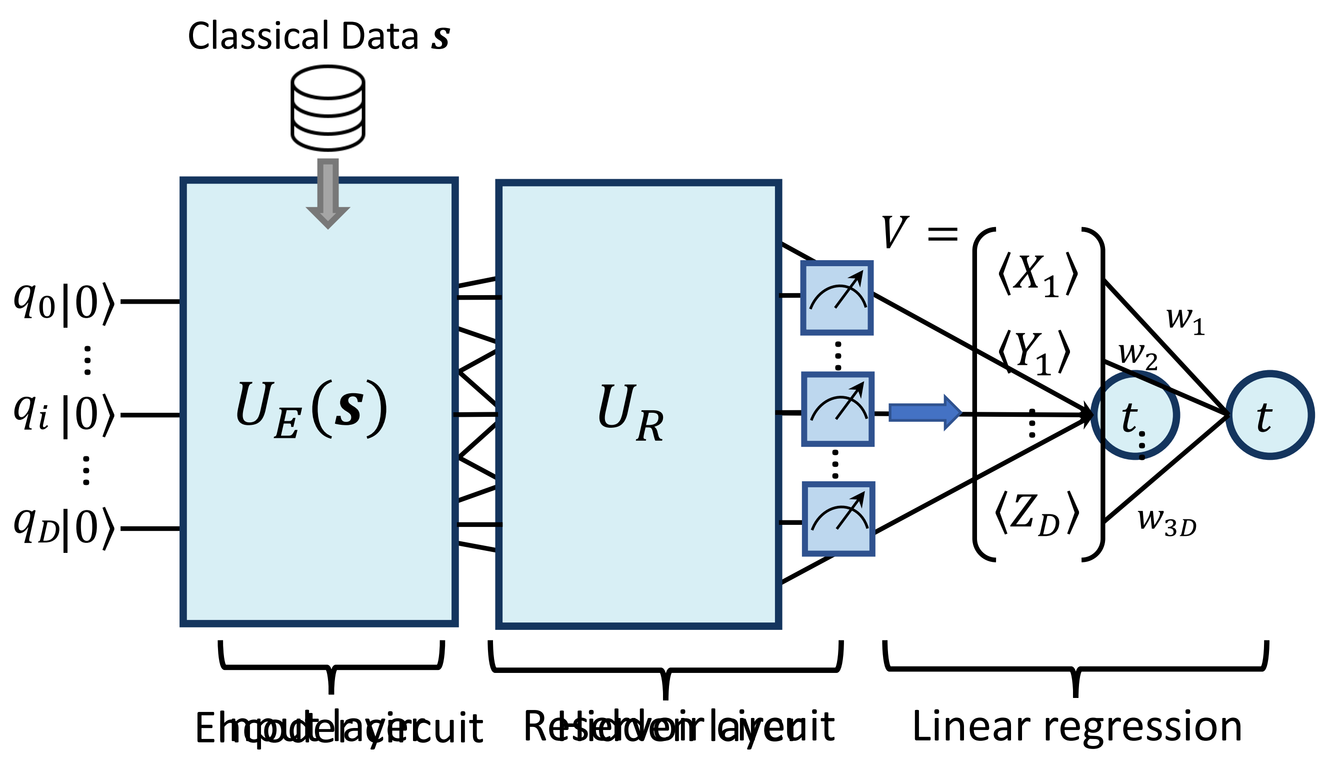


Background – Extreme Learning Machine (ELM)



- Feedward neural network
- Feed classical data into input layer
- Hidden layer: **fixed** and **randomly** assigned weights and biases
- Train the **linear regression model** on the output layer's weights to predict the target value

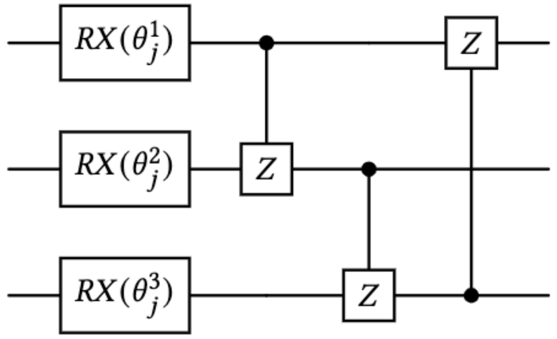
Background – Quantum Extreme Learning Machine



- Replace neural into quantum circuit
- Feed classical data into encoder circuit to transfer into quantum states
- Output state of the encoder goes into a quantum reservoir circuit, whose parameters are fixed and randomly assigned
- A set of observables are applied to obtain the output vector of measured values
- Train the **linear regression model** on the output layer's weights to predict the target value

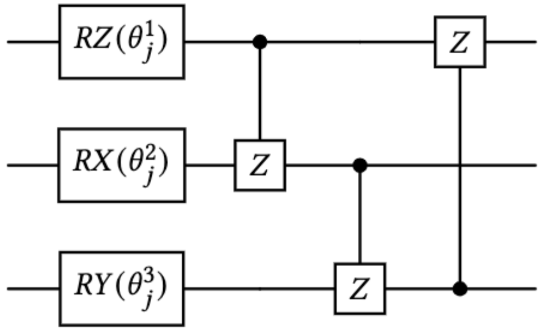
QUELL – Quantum Extreme Learning Machines (QELM)

- **Determined hardware efficient encoder (DHE):**
  - Each qubit is applied with an RX
  - The rotation value θ of RX is determined by the input value
  - CZ gates are applied with a cyclic entangling structure to connected all qubits



Determined hardware efficient **(DHE) encoder**

- **Random hardware efficient encoder (RHE):**
  - Each qubit is applied with a randomly selected rotation gates from $\{\mathrm{RX}, \mathrm{RY}, \mathrm{RZ}\}$
  - The rotation value θ is determined by the input value
  - CZ gates are applied with a cyclic entangling structure to connected all qubits



Randomized hardware efficient **(RHE) encoder**

QUELL – Encoder Types

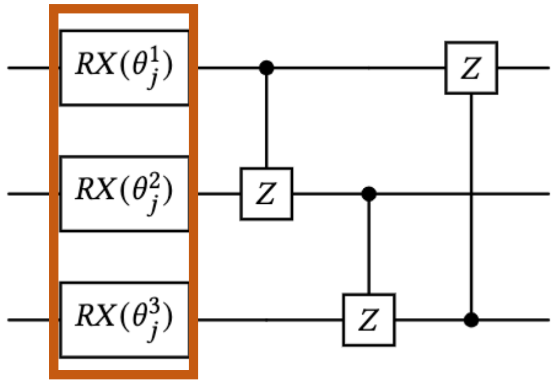
Features, e.g., number of upward calls

Passenger Traffic Data

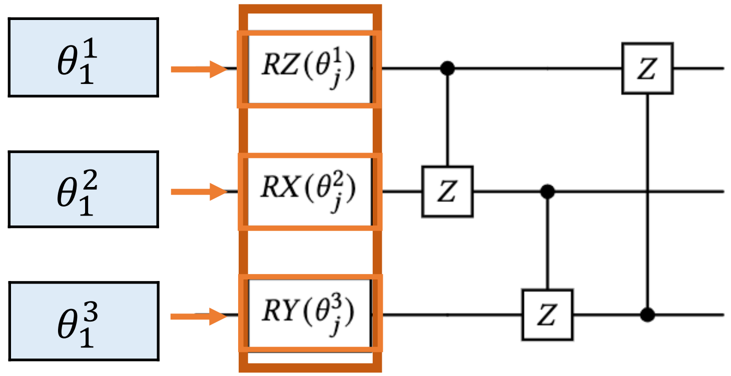
Time	F_1	F_2	F_3
tw_1	s_1^1	s_1^2	s_1^3
…	…	…	…
tw_P	s_P^1	s_P^2	s_P^3

Time window of 5 min

Time	F_1	F_2	F_3
tw_1	θ_1^1	θ_1^2	θ_1^3
…	…	…	…
tw_P	θ_P^1	θ_P^2	θ_P^3



Determined hardware efficient **(DHE) encoder**



Randomized hardware efficient **(RHE) encoder**

- Encoders process input data by parameterizing **Rotation Gate**.
- To avoid multiple values being encoded in the same state, min-max **normalization** is used to scale feature values to a range of **0 to π radians.**
- Feed normalized feature values of **each time window** into quantum encoder

QUELL – Reservoir Types

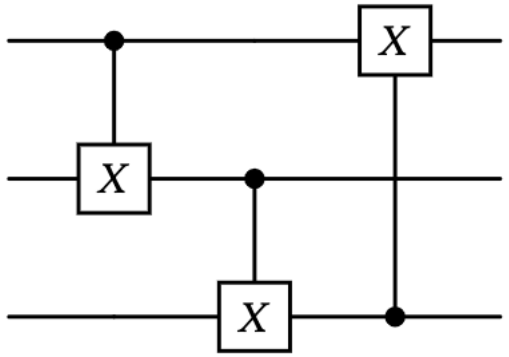
- **CNOT reservoir (CNOT):**
  - Apply CNOT gates in cyclic entangling structure
- **Harr random reservoir (Harr):**
  - Apply a single unitary sampled from the Harr measure
- **Ising Mag Traverse Reservoir (Ising):**
  - Made up of a single Ising Hamiltonian with random coefficients
  - Unitary operator:

$$U_{ISING} = \exp(-iH\Delta t)$$

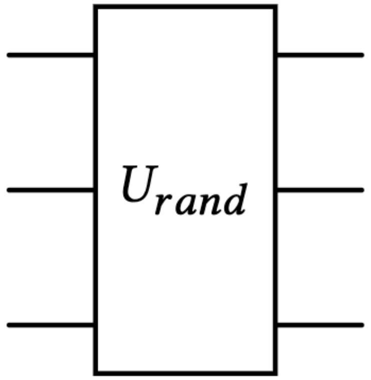
  - Hamiltonian (coefficients J and a are randomly selected):

$$H = \sum_{k,j} J_{k,j} Z_k Z_j + \sum_j a_j X_j$$

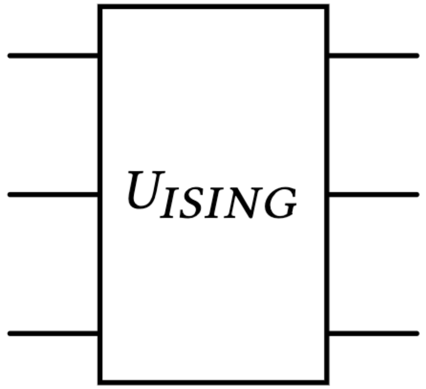
- **Rotation Reservoir (Rotation):**
  - Each qubit is applied with a randomly selected rotation gates from {RX, RY, RZ}
  - The rotation value α is randomly selected
  - CX gates are applied with a cyclic entangling structure



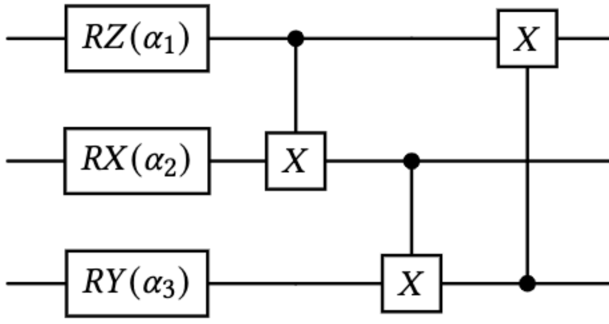
CNOT reservoir



Harr random
(Harr) reservoir

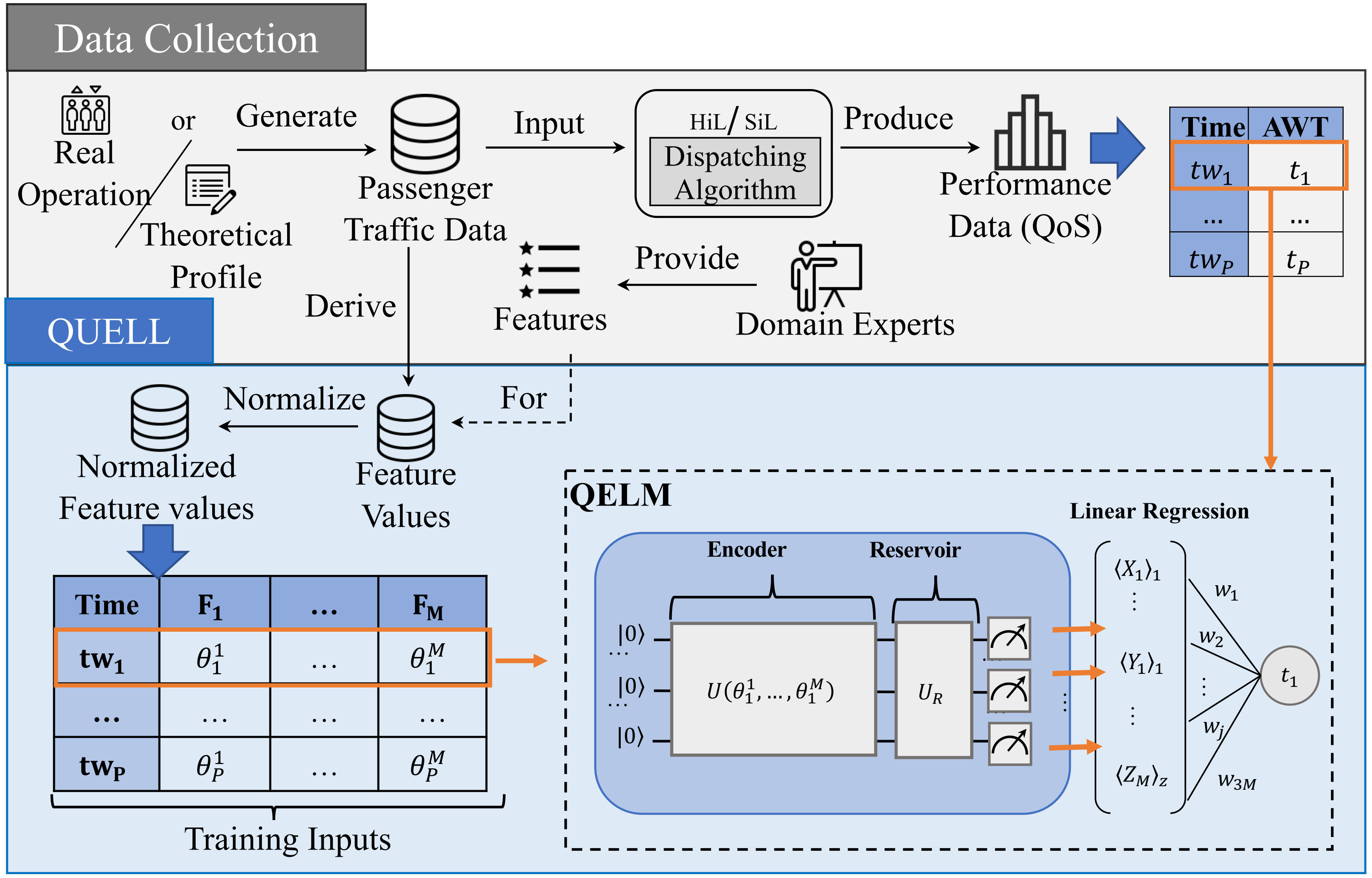


Ising Mag Traverse
(ISING) reservoir



Rotation reservoir

QUELL – Overview



Research Questions

- **RQ1:** Which **combination** of **encoder and reservoir** of QUELL achieves the best prediction performance with a different number of features?
- **RQ2:** Using **the optimal combination** of encoder and reservoir, what is the minimum number of features for which QUELL achieves a prediction performance comparable to that using the maximum number of features?
- **RQ3:** How well does QUELL perform compared to the baseline when using different numbers of features for predictions?

Experiment Design

Datasets

- 4 days **passenger traffic data** extracted from **real operation** of elevators installed in a 10-floor building with time window of **5 min:** $ExpDay_1$, $ExpDay_2$, $ExpDay_3$ and $ExpDay_4$

Features

F	Description	F	Description
F_1	Number of upward calls from low-level floors.	F_7	Average distance of the travel from the upward calls.
F_2	Number of upward calls from medium-level floors.	F_8	Average distance of the travel from the downward calls.
F_3	Number of upward calls from high-level floors.	F_9	Number of total upward calls in the past 5 minutes.
F_4	Number of downward calls from low-level floors.	F_{10}	Number of total downward calls in the past 5 minutes.
F_5	Number of downward calls from medium-level floors.	F_{11}	Number calls going upwards.
F_6	Number of downward calls from high-level floors.	F_{12}	Number calls going downwards.

Feature Sets

FS	Selected
FS_2	F_{11}, F_{12}
FS_{3a}	F_{11}, F_{12}, F_7
FS_{3b}	F_{11}, F_{12}, F_1
FS_4	F_{11}, F_{12}, F_7, F_8
FS_5	$F_{11}, F_{12}, F_7, F_8, F_1$
FS_{10}	$F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}$

- **QoS metric:** average waiting time (AWT) generated by elevator simulator *ELEVATE*

Experiment Design

Baseline:

- DARIO$_{\text{PRED}}$[3] with SVM
- DARIO$_{\text{PRED}}$ with Regression Tree

Quantum environment

Quantum framework and ideal simulator:

- Qreservoir package
- Qulacs framework

Evaluation metrics

- Mean square error (MSE) of predicted AWT time:

$$MSE = \frac{1}{P}\sum_{j=1}^{P}(t_j^{pre} - t_j)^2$$

- We repeat each experiment 30 times to reduce the randomness, thus, we will also calculate the average MSE value.

$$AMSE = \sum_{i=1}^{30} MSE_i/30$$

Statistical tests

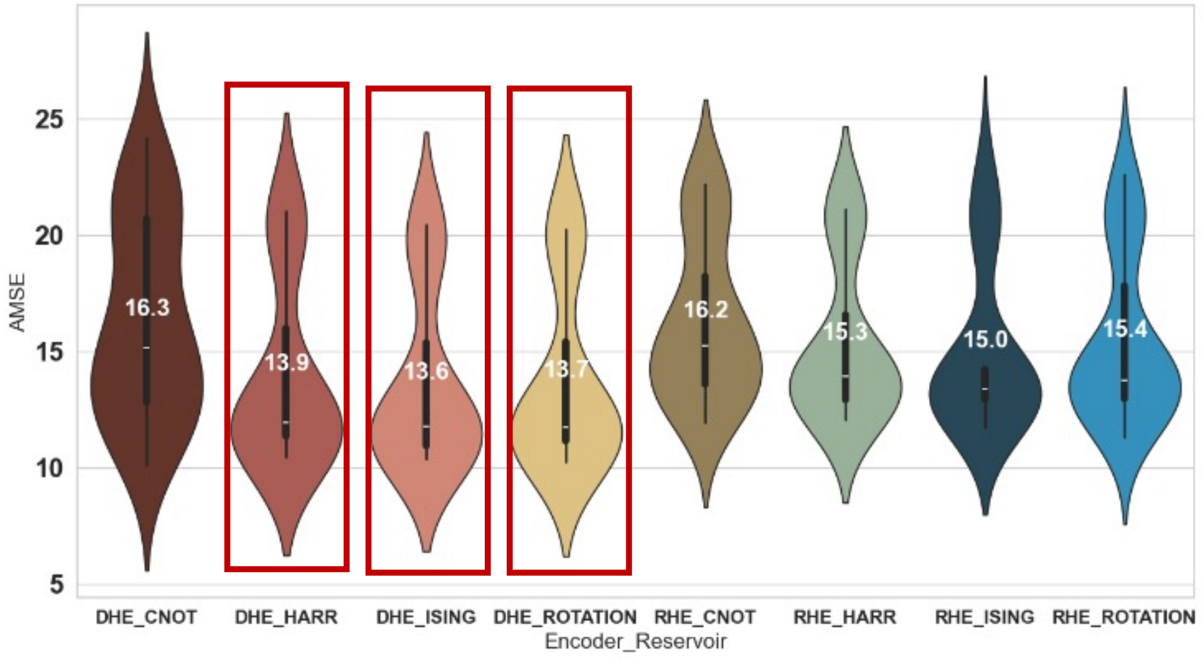
- Mann-Whitney U test with $\hat{A}_{12}$ effect size
- One-sample Wilcoxon test with Cohen's *d* to interpret magnitude

[3] Aitor Gartziandia, Aitor Arrieta, Jon Ayerdi, Miren Illarramendi, Aitor Agirre, Goiuria Sagardui, and Maite Arratibel. 2022. Machine learning-based test oracles for performance testing of cyber-physical systems: An industrial case study on elevators dispatching algorithms. Journal of Software: Evolution and Process 34, 11 (2022), e2465. https://doi/10.1002/smr.2465

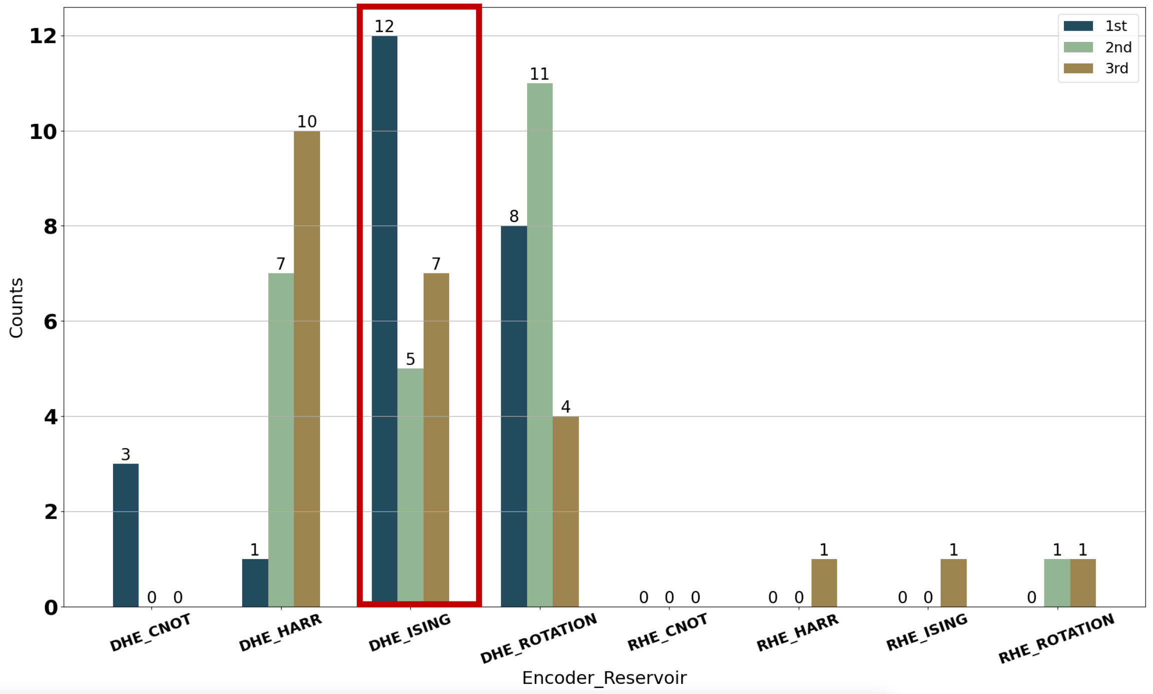
RQ1: Which combination is the optimal?

Selecting the optimal *encoder_reservoir* combination

Violin plot of $AMSE$ values of 24 settings (6 features sets and 4 datasets)



1st, 2nd, and 3rd position of each combination for 24 settings



Encoder_Reservoir	1st	2nd	3rd
DHE_CNOT	3	0	0
DHE_HARR	1	7	10
DHE_ISING	12	5	7
DHE_ROTATION	8	11	4
RHE_CNOT	0	0	0
RHE_HARR	0	0	1
RHE_ISING	0	0	1
RHE_ROTATION	0	1	1

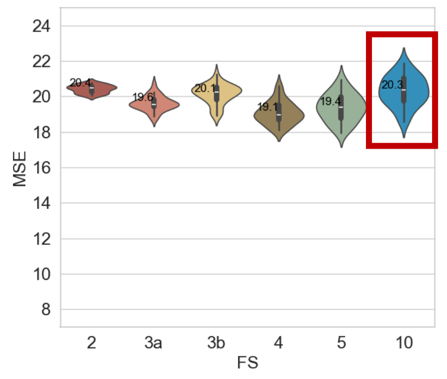
Overall, the **ISING reservoir** combined with the **DHE encoder** enables QUELL to perform the best.

RQ2: Which number of features is comparable to maximum

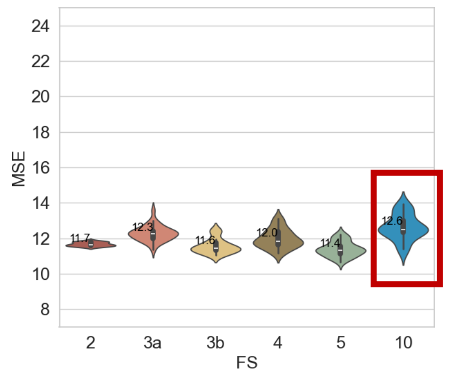
QUELL's performance with the best setting on different feature sets

Violin plot of MSE values of 30 runs

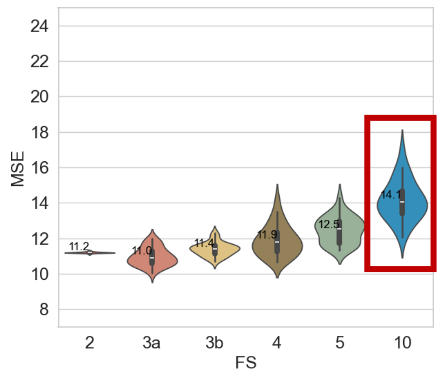
$ExpDay_1$



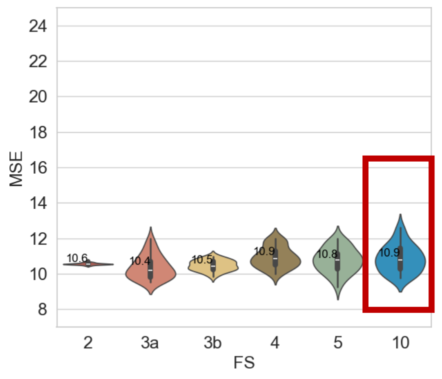
$ExpDay_2$



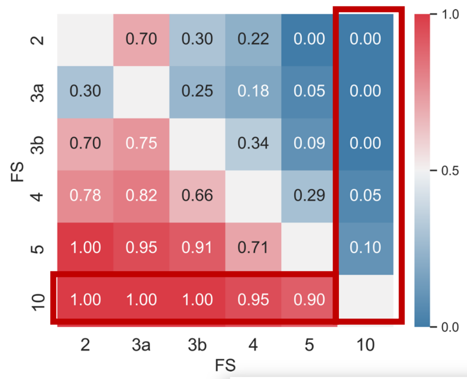
$ExpDay_3$

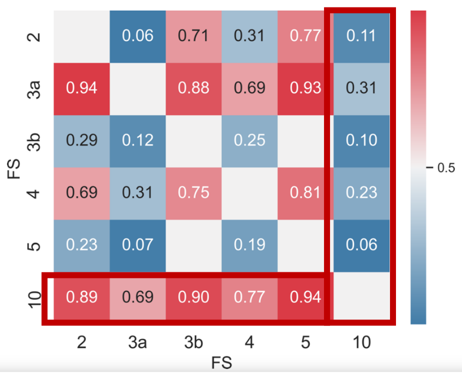


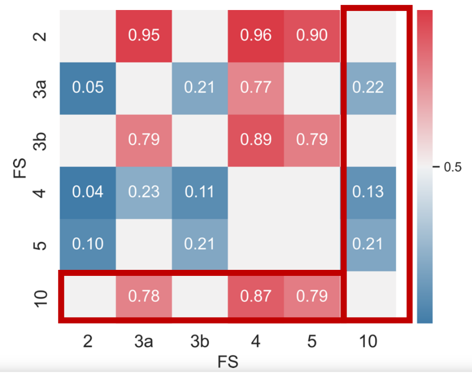
$ExpDay_4$

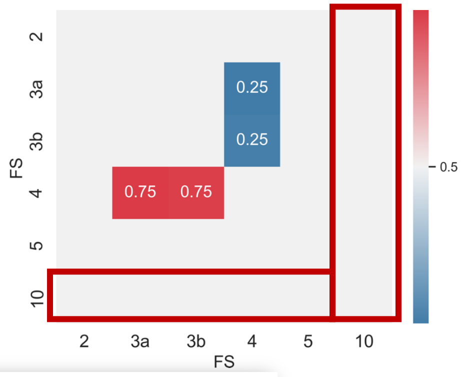


Comparison of QUELL with different feature sets









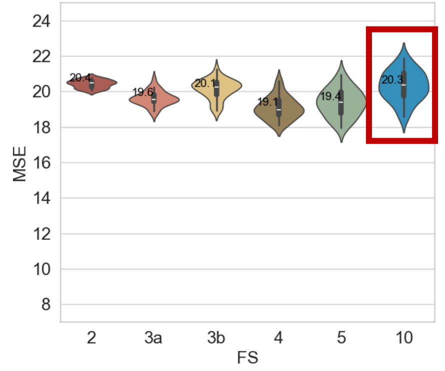
Overall, QUELL with **few features outperforms** QUELL with the maximum number of features 10. This shows the effectiveness of QELM in our industrial context.

RQ2: Which number of features is comparable to maximum

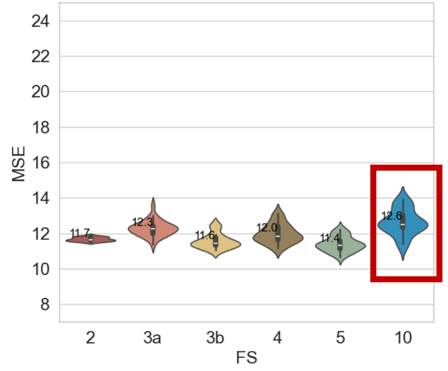
QUELL's performance with the best setting on different feature sets

Violin plot of MSE values of 30 runs

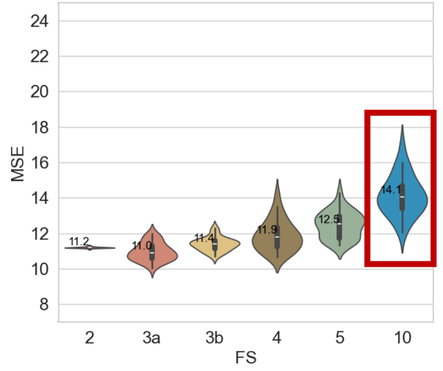
$ExpDay_1$



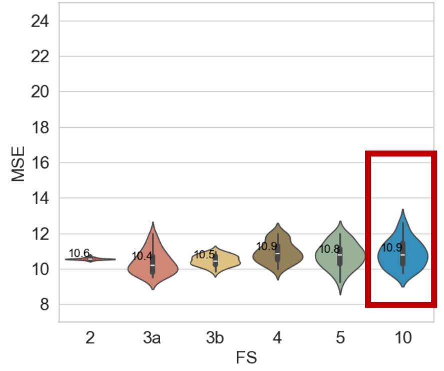
$ExpDay_2$



$ExpDay_3$



$ExpDay_4$

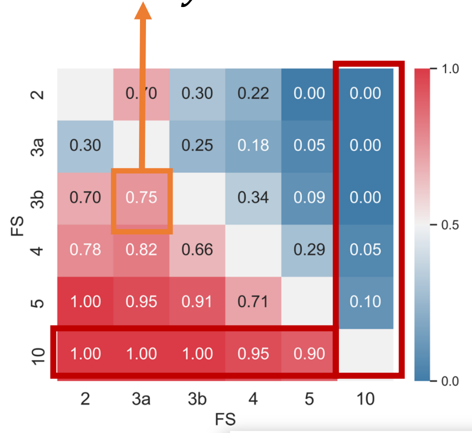


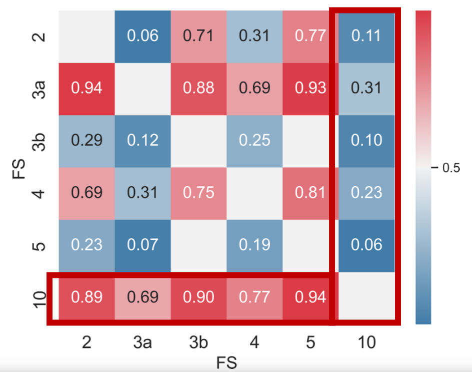
x axis$>$ y axis

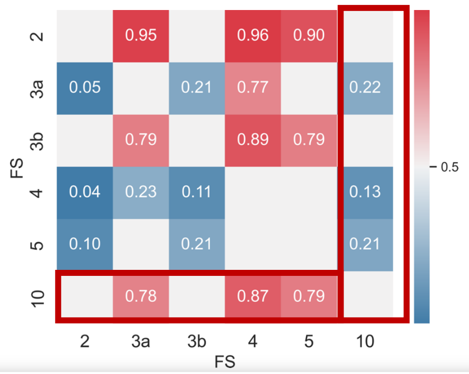
Comparison of QUELL with different feature sets

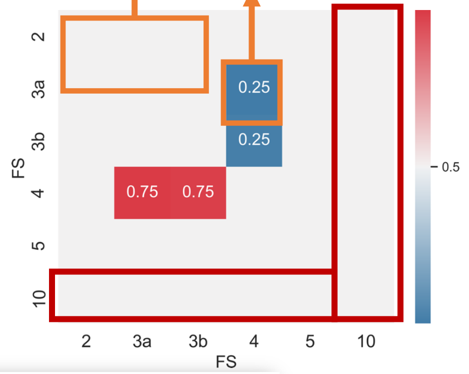
x axis$\equiv$ y axis

x axis$<$ y axis









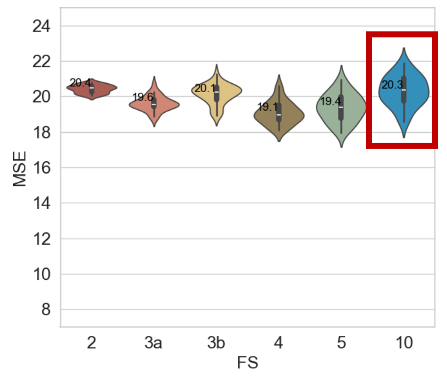
Overall, QUELL with **few features outperforms** QUELL with the maximum number of features 10. This shows the effectiveness of QELM in our industrial context.

RQ2: Which number of features is comparable to maximum

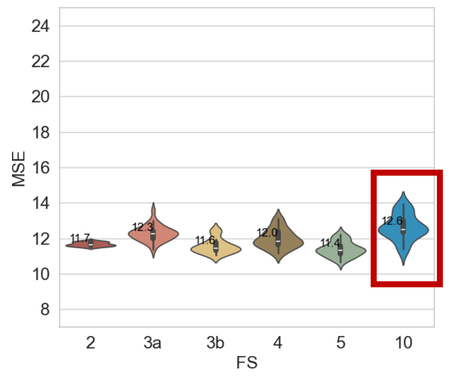
QUELL’s performance with the best setting on different feature sets

Violin plot of MSE values of 30 runs

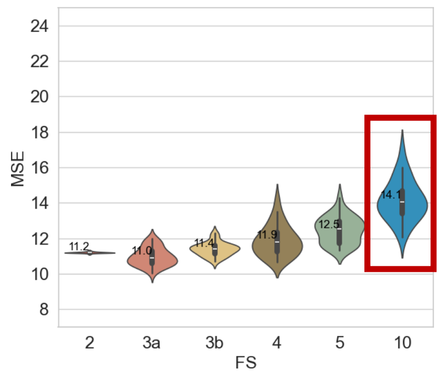
$ExpDay_1$



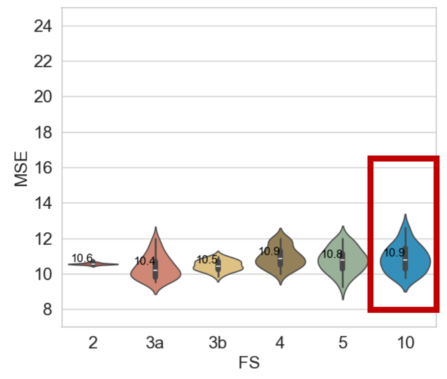
$ExpDay_2$



$ExpDay_3$



$ExpDay_4$

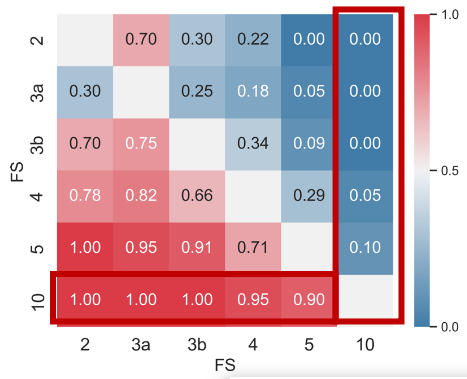


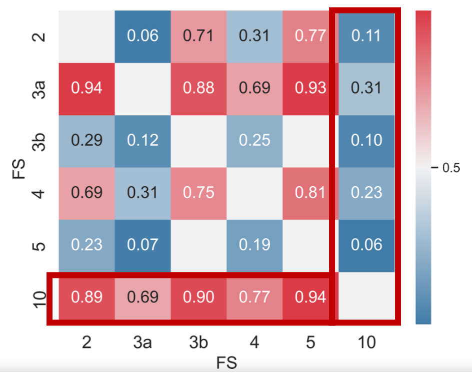
Comparison of QUELL with different feature sets

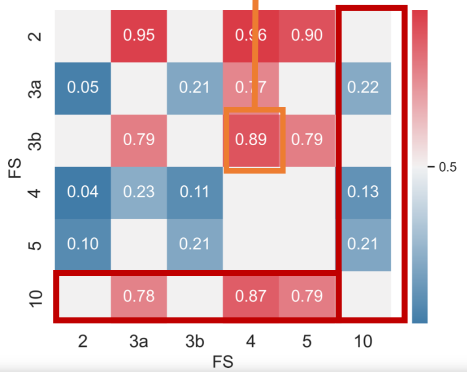
x axis$>$ y axis

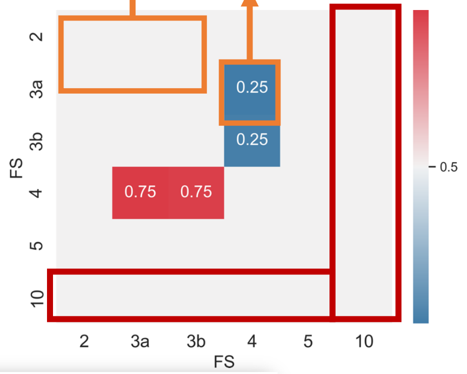
x axis$\equiv$ y axis

x axis$<$ y axis









Overall, QUELL with **few features outperforms** QUELL with the maximum number of features 10. This shows the effectiveness of QELM in our industrial context.

RQ3: How well does QUELL perform compared to the baseline

- We perform $\text{DARIO}_{\text{PRED}}$ with SVM and regression tree and QUELL on 6 feature sets of 4 datasets
- We perform a **one-sample Wilcoxon signed rank test** to compare MSE values of $\text{DARIO}_{\text{PRED}}$ with QUELL
  - All p-values are lower than 0.05
  - Results indicates significant difference between QUELL with all feature sets and datasets with two classical algorithms.
- We compute Cohen's *d* effect size to see the magnitude of differences
  - All calculated *d* values are lower than -1
  - MSE values generated by QUELL are greatly smaller than that generated by $\text{DARIO}_{\text{PRED}}$

For the same prediction task in our industrial context, QUELL **outperforms** classical machine learning approaches. This demonstrates the potential of QELM

Lessons Learned

Potential Applications

- Run-time prediction
- Building digital twins
- Prediction problems in other contexts

Research Implications

- Classical and quantum software engineering
- Empirical study

Future Work

- Involve hardware noise
- Further configuration of encoders and reservoirs

Conclusion

- An industrial application of quantum extreme learning machine (QELM) for solving the waiting time prediction task in the context of elevator

- Four real datasets from the elevators’ real operation

- QELM could offer benefits by performing significantly better prediction even with fewer features

Motivation – QELM for Software Testing in Practice

Ideal simulations do not reflect the reality of with noise

Simulating large-scale industrial problems is unfeasible

QELM in real-world applications with the noise is rarely explored

Motivation

- Examining the impact of **quantum noise** on QELM models through three **industrial, real-world** case studies[4]
- Assessing the feasibility of combining QELMs with **noise error mitigation** techniques to enhance their applicability

[4] Muqeet, Asmar, et al. "Assessing Quantum Extreme Learning Machines for Software Testing in Practice." arXiv preprint arXiv:2410.15494 (2024).

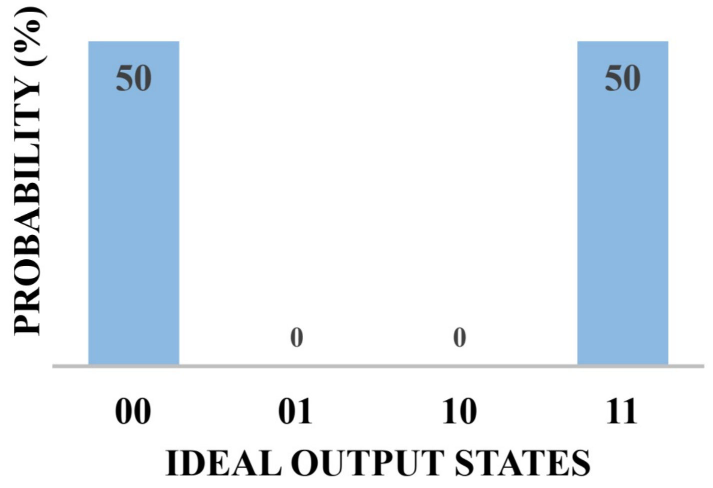
Background

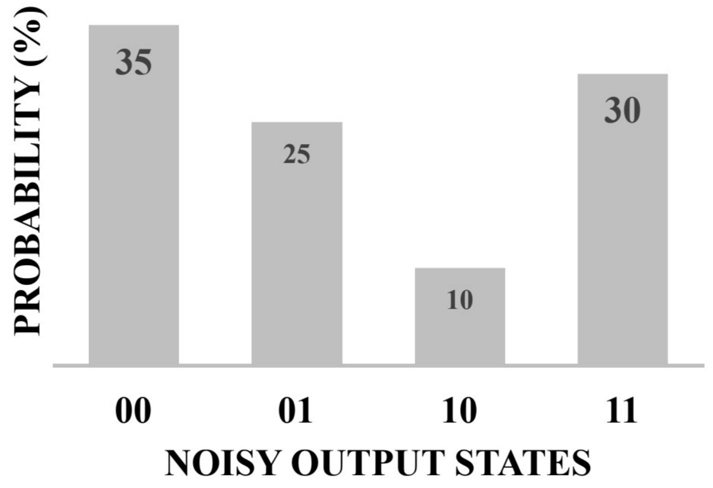
Source of quantum noise

- **Decoherence**
  Interactions between qubits and environments lead to disturbances and loss of information in quantum states

- **Crosstalk noise**
  Unwanted interactions between qubits leads to unintended quantum states

- **Hardware calibration error**
  - Minor calibration errors can result in slight lead to undesirable states following a series of gate operations





Error Mitigation Methods

- **Non-ML error mitigation: Zero Noise Extrapolation (ZNE)**[5]
  - Step 1: Intentionally scale noise by methods such as applying additional gates
  - Step 2: Extrapolate to zero noise with mathematical approaches

- **ML error mitigatioin: Q-LEAR**[6]
  - Step 1: Extract circuit level features and output level features
  - Step 2: Train a ML noise model based on the features and ideal outputs

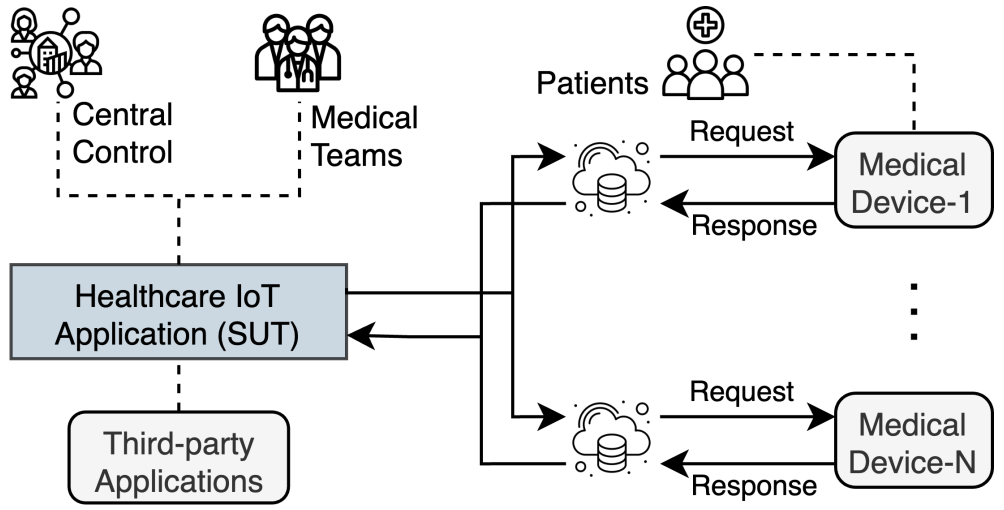
[5] LaRose, Ryan, et al. "Mitiq: A software package for error mitigation on noisy quantum computers." Quantum 6 (2022): 774.

[6] Muqeet, Asmar, et al. "A machine learning-based error mitigation approach for reliable software development on IBM's quantum computers." Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering. 2024.

Case Study – Oslo City Healthcare Data

- Oslo City provides healthcare services to its residents
- An Healthcare IoT-based platform connects medical devices with pharmacies, caregivers, patients
- Midical devices are allocated to patients to enable real-time alerts and personalized care

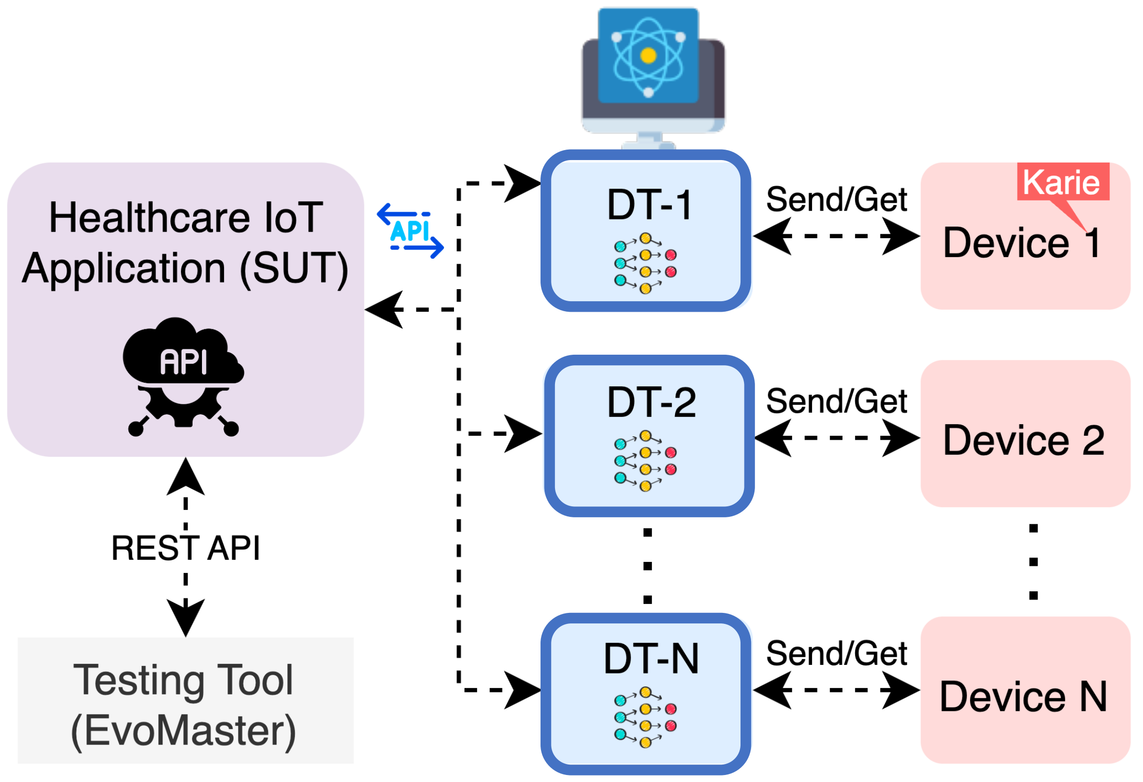
Challenge: System-level testing of IoT healthcare applications involves multiple medical devices, but using them in tests leads device damage or service blocking



Karie midical dispenser

Case Study – Oslo City Healthcare Data

- ML-based digital twins (DTs) are proposed to facilitate the automated and thorough testing[7]
- A testing tool generates REST API tests, and SUT communicates with DTs that manage all API calls
- Based on the dataset for building DTs of *Karie*, we train QELM model to predict responses (HTTP status codes) to support automated testing



Inputs: 18 features, such as brightness setting, language preference, alarm configurations, network connectivity[4]
Outputs: HTTP states codes

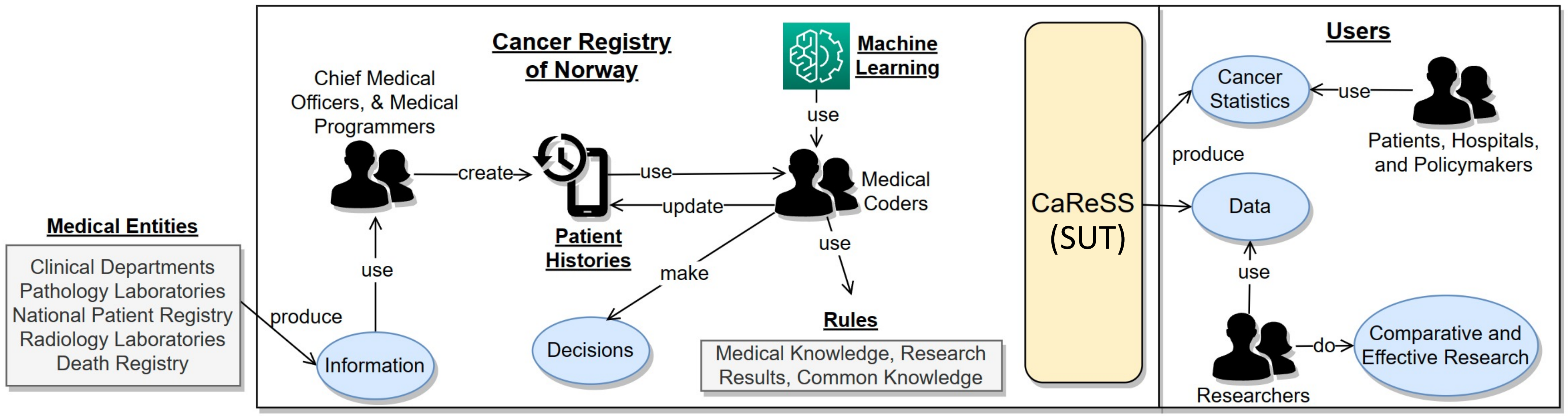
Devices only used for calibration or synchronization

[7] H. Sartaj, S. Ali, and J. M. Gjøby, “MeDeT: Medical Device Digital Twins Creation with Few-shot Meta-learning,” 2024. [Online]. Available: https://arxiv.org/abs/2410.03585§

Case Study – Norway’s Cancer Registry Data

- The Cancer Registry of Norway (CRN) collects cancers cases across Norway by receiving *cancer messages* from health institutes
- Cancer messsages are validated with a set of rules by an automated Cancer Registration Support System (CaReSS)
- CaReSS also analyzes the collected data and generates statistics for policymakers and healthcare stakeholders

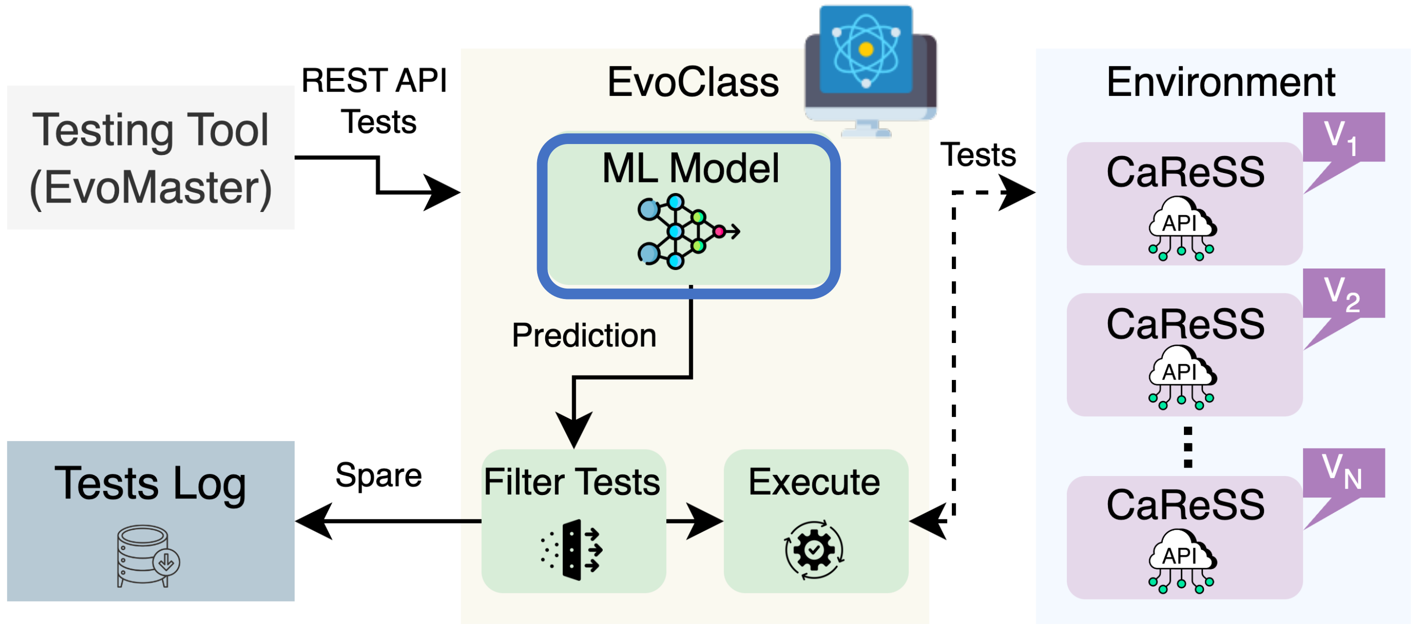
Challenge: When testing CaReSS, each request is running in real-time, executing invalid requests incurs unnecessary execution costs and impacts performance of CaReSS during operation



Case Study – Norway’s Cancer Registry Data

- A testing tool generates REST API tests, and an ML-based approach, EvoClass, is proposed to filter test cases likely to be invalid[8]

- Based on the CaReSS rule engine dataset, we train a QELM model to predict potentially successful or unsuccessful tests



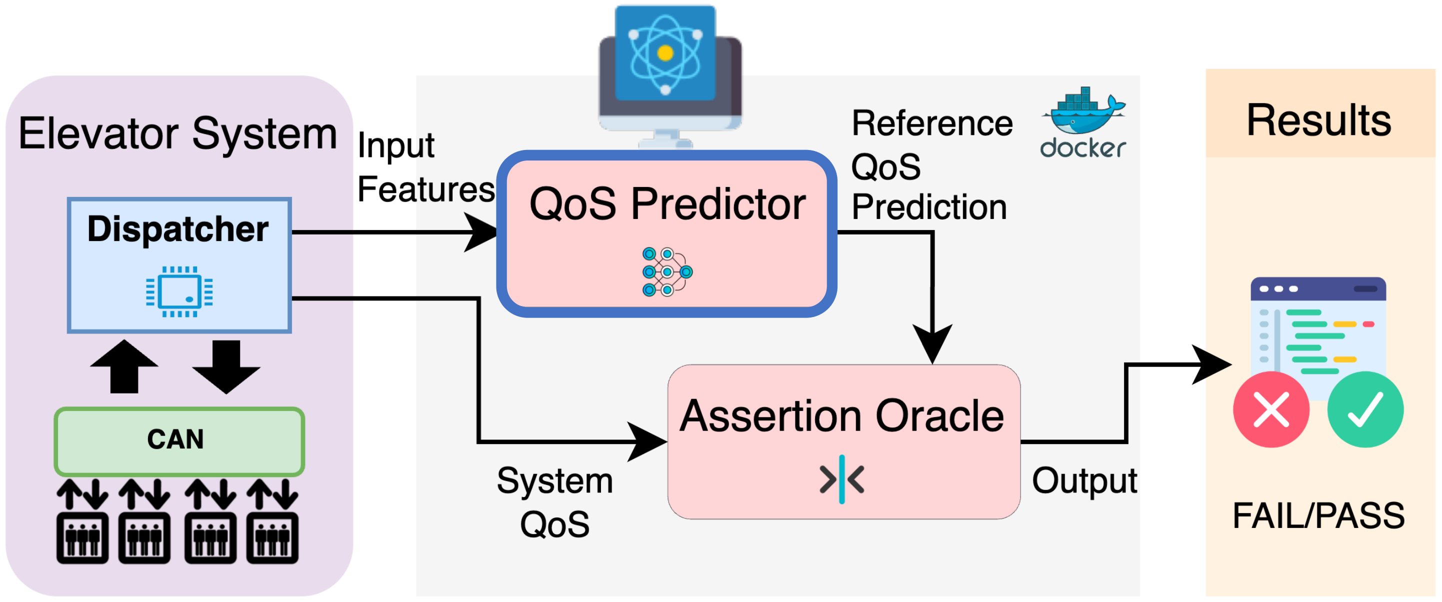
Inputs: 57 features such as patient medical records, cancer type, tumor behavior.
Output: success/failure

[8] Isaku, Erblin, et al. "Cost Reduction on Testing Evolving Cancer Registry System." 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2023.

Case Studies – Orana Elevator

- Based on passenger information, we train a QELM model to predict the passenter average waiting time

Inputs: 12 features
Output: average waiting time in 5 min



Research Questions

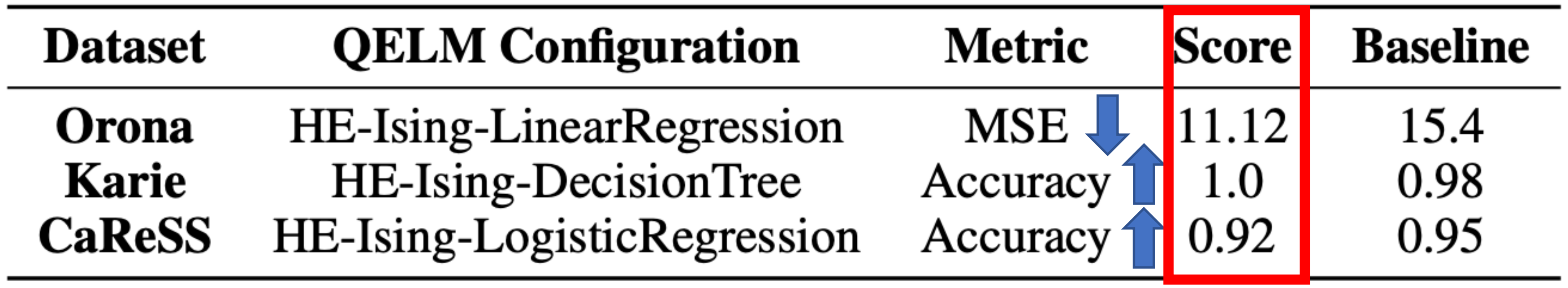
- **RQ1:** How resistant is QELM to quantum noise?
- **RQ2:** How effective are current practical error mitigation methods for QELMs?

Experiment Setting

- **Features:**
  - Select key features based on feature importance scores
  - Orana dataset: 3 features; Karie dataset: 4 features; CaReSS dataset: 8 features

- **Noise model:**
  - IBM Sherbrook
  - IBM Torino
  - IBM Fez

Optimal QELM configuration under ideal simulation comparing with classical baselines

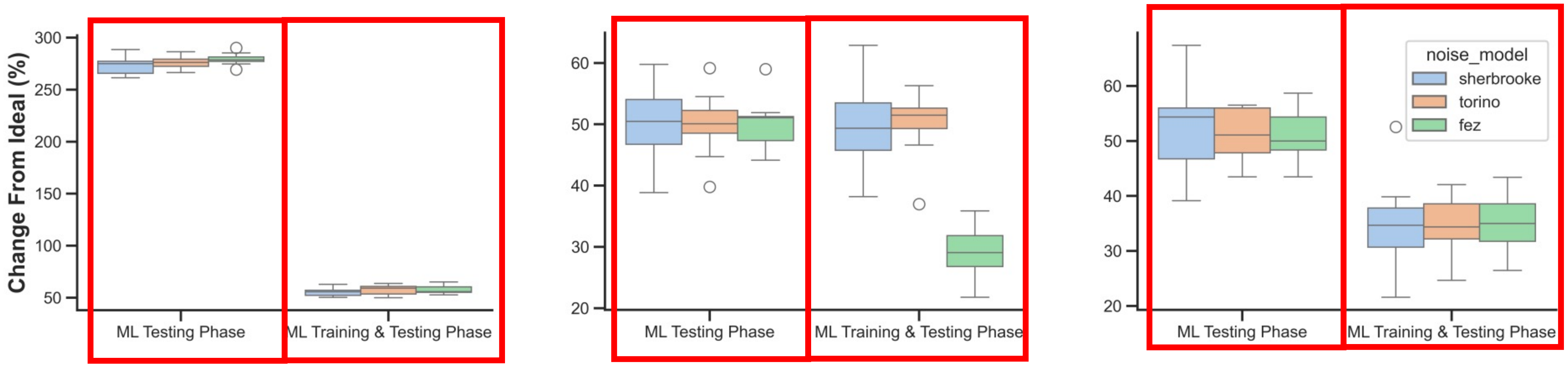


Dataset	QELM Configuration	Metric	Score	Baseline
Orona	HE-Ising-LinearRegression	MSE ↓	11.12	15.4
Karie	HE-Ising-DecisionTree	Accuracy ↑	1.0	0.98
CaReSS	HE-Ising-LogisticRegression	Accuracy ↑	0.92	0.95

Resistance to Quantum Noise

Adding noise only training phase

Adding noise both training and testing phases

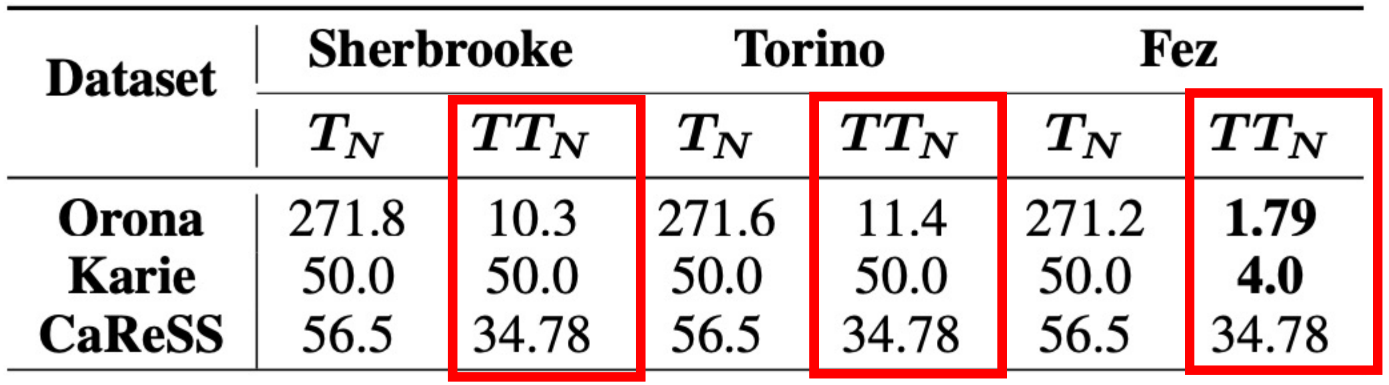


(a) Orona Dataset (b) Karie Dataset (c) CaReSS Dataset

- Noise in both training and testing phases reduces its impact on model performance
- Error mitigation is required for practical use

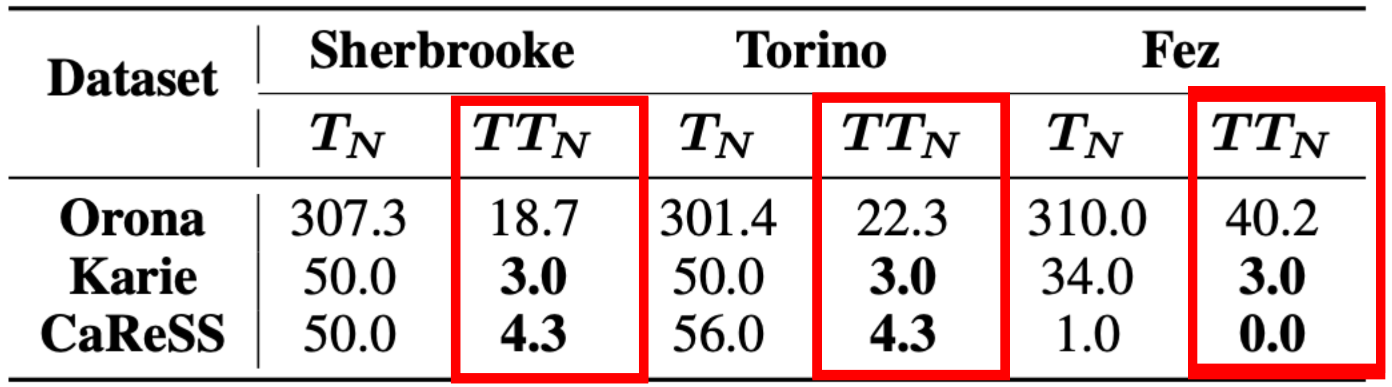
Integration with Error Mitigation

Integration with ZNE



Dataset	Sherbrooke		Torino		Fez	
	T_N	TT_N	T_N	TT_N	T_N	TT_N
Orona	271.8	10.3	271.6	11.4	271.2	**1.79**
Karie	50.0	50.0	50.0	50.0	50.0	**4.0**
CaReSS	56.5	34.78	56.5	34.78	56.5	34.78

Integration with Q-LEAR



Dataset	Sherbrooke		Torino		Fez	
	T_N	TT_N	T_N	TT_N	T_N	TT_N
Orona	307.3	18.7	301.4	22.3	310.0	40.2
Karie	50.0	**3.0**	50.0	**3.0**	34.0	**3.0**
CaReSS	50.0	**4.3**	56.0	**4.3**	1.0	**0.0**

- ZNE are constrained by qubit size and noise models
- ML-based methods like Q-LEAR excel in classification tasks but struggle with regression
- Integrating error mitigation methods enhances the noise resistance of QELMs, but their effectiveness is context-dependent.

Values show the median (among 10 repeats) percentage change from the ideal values
Bold values show significant improvement by error mitigation

Adding error mitigation both training and testing phases

Lessons Learned

QELM Application

- Potential to outperform classical machine learning models
- Scalability issues due to quantum noise and qubit limitations, requiring solutions

Practical Limitations

- Real quantum computers and effective error mitigation techniques required for larger problems, which may introduce significant computational overhead
- QELMs with tailored error mitigation strategies may be valuable for specialized fields

Conclusion

- This paper evaluated the practical application of QELMs under realistic quantum noise conditions across three industrial case studies in classical software testing
- QELMs perform well in ideal simulations; however, they are affected by quantum noise
- Error mitigation techniques can enhance noise resistance, and tailored error mitigation strategies for QELM are needed to enhance their applicability

Acknowledgements

Collaborators

Shaukat Ali

Paolo Arcaini

Aitor Arrieta Marcos

Asmar Muqeet

Hassan Sartaj

Maite Arratibel

Julie Marie Gjøby

Narasimha Raghavan Veeraragavan

QUELL[1]

QELM in Practice[2]

[1] Wang, Xinyi, et al. "Application of quantum extreme learning machines for qos prediction of elevators' software in an industrial context." *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 2024.
[2] Muqeet, Asmar, et al. "Assessing Quantum Extreme Learning Machines for Software Testing in Practice." *arXiv preprint arXiv:2410.15494* (2024).

UNIVERSITY OF OSLO

NII 国立情報学研究所 National Institute of Informatics

MONDRAGON

simula OSLOMET Orona

NIPH Norwegian Institute of Public Health

E-mail: xinyi@simula.no