Day - Task Sheet

Topics:

1. **Quantum data encoding & feature mapping**
2. **Variational circuits & cost-function optimization**
3. **Quantum classifiers training**
4. **Gradient-based training in hybrid systems**

Description:

This worksheet guides students through

- load and preprocess a real-world dataset and map it to qubits.
- compare feature maps and variational circuits under different depths/entanglement
- implement QSVC and QNN classifiers
- train with gradient-free (SPSA/COBYLA) and gradient-based methods
- evaluate when/why quantum models can show advantages in learning tasks

Requirement:

Part A — Dimensionality Reduction & Qubit Encoding
Objective: Reduce the feature space so that the number of components equals the number of qubits. Normalize to a range suitable for angle/phase encodings (e.g., $[0, \pi]$).

• Download and Load the NGSIM Dataset

Download the NGSIM dataset (Next Generation Simulation). Suggested portal: https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajector/8ect-6jqj

Choose a manageable subset (e.g., I-80 or US-101 trajectory data).

Load with pandas and inspect key fields such as Vehicle_ID, Frame_ID, Local_X, Local_Y, v_Vel, v_Acc, Lane_ID.

Optionally sample or filter to a balanced binary classification task (e.g., lane-change vs. no lane-change), or define your own label.

• Apply PCA to obtain reduced features with n_components $\in \{4, 6, 8, 10\}$; alternatively, train a simple Autoencoder (AE) to compress to these dimensions

• Standardize or min-max scale features before dimensionality reduction

• For PCA, plot explained variance ratio to justify component choices

• Map reduced feature vectors to qubits 1:1 (components → qubits)

Please use only 200 training and 50 testing samples.

Part B — Quantum Feature Mapping

Objective: Embed the reduced classical features into quantum states using different feature maps. Evaluate how depth and entanglement influence performance

• Choose the type of feature map

ZFeatureMap: basic Z-rotation encoding.
(Qiskit: ZFeatureMap(feature_dimension=n_qubits))

ZZFeatureMap: adds pairwise ZZ entanglement.
(Qiskit: ZZFeatureMap(feature_dimension=n_qubits))

PauliFeatureMap: composite encoding with chosen Paulis (e.g., X, Y, Z).
(Qiskit: PauliFeatureMap(feature_dimension=n_qubits, paulis=["X","Y","Z"]))

Part C — Implement QSVC or QNN

Objective: Implement the QML models.

Sub-task – QSVC: Use qiskit_machine_learning.algorithms.QSVC with the chosen feature map and an appropriate backend.

Sub-task – QNN: Implement a SamplerQNN or EstimatorQNN with a simple variational ansatz (e.g., EfficientSU2) and a classical optimizer.

• After initilize the model, train on each qubit setting (4, 6, 8, 10) using the same train/test split for fair comparison

Part D — Ideal and Depolarising Noise Environments (AerSimulator)

Objective: Run each model under both ideal and noisy simulators and compare results.

• Ideal simulator: AerSimulator()

• Depolarising noise: construct a NoiseModel and add depolarising_error on relevant instructions (e.g., u, cx).

Part E — Results and Visualisation.

Objective: Clearly present the visualisation.

Required plots

1. Plot Accuracy vs Number of Qubits (4, 6, 8, 10).
2. Compare Accuracy (Ideal vs Noisy) for each feature map and model type.

3. Optionally compare training/inference time across settings.
4. Summarise findings: impact of qubit count, feature map choice, and noise on performance.

Reflection

• Compare QSVC vs QNN: when does each do better on NGSIM?

• Explain how encoding choice (Z/ZZ/Pauli) and entanglement affect expressivity vs runtime

• Discuss gradient-free vs gradient-based training: stability and speed on this task.

• Describe how noise and finite shots influence accuracy

Submission Details:

1. All files are required to be submitted, Jupyter-Notebook (.ipynb)
2. Send email to quantumlab.deakin@gmail.com with registered email and subject "Day-3 Firstname_Lastname"