Experimental Setup: Where Parameters Are Defined

QCardEst/QCardCorr

December 20, 2025

This document shows exactly where each experimental setup parameter is defined in the codebase.

1 Overview

The experimental setup parameters are configured in the `settings` **dictionary** in `runRegression.py`, which can be:

1. **Set directly** in the file (lines 14-36)
2. **Overridden via command line** (lines 38-40)
3. **Automatically determined** from data (for some parameters)

2 Query Characteristics: Up to 6 Joined Tables

2.1 Where Defined:

- **Primary**: `cardEnv.py`, line 50
- **Source**: Automatically extracted from data file

```
# cardEnv.py, line 48-50
queryInfo = extractTablesFromCardsCSVfile(file)
self.setMaxId(queryInfo["nTables"] - 1)
self.numFeatures = queryInfo["maxQuerySize"]  #     Up to 6 tables
```

Data Source:

- `costs/stats/statsCards6.csv` → max 6 tables
- `costs/stats/stats4.csv` → exactly 4 tables
- `costs/jobSimple/job.csv` → variable

Manual Override:

- `runRegression.py`, line 35: `"numFeatures": 6` (can be set manually)

3 Quantum Model: 6 Qubits

3.1 Where Defined:

- **Primary**: `ML/models.py`, lines 39-41
- **Calculation**: `nQubits = max(nInputs, nQubitsOut)`

```
# ML/models.py, lines 39-41
nQubitsOut = math.ceil(math.log2(nOutputs))  # Output qubits needed
nQubits = max(nInputs, nQubitsOut)  #     6 qubits (from numFeatures=6)
```

Determined by:

- `nInputs` = `env.getInputSize()` = `numFeatures` = 6
- `nOutputs` = `agent.nInputs` (depends on interpretation layer, typically 2)

Result: With `numFeatures=6` and typical `nOutputs=2`, we get `nQubits = max(6, 1) = 6`

4 VQC Layers: 16 Layers

4.1 Where Defined:

- **Primary**: `runRegression.py`, line 18 (default: 6, but overridden to 16)
- **Used in**: `ML/circuits.py`, line 86

```
# runRegression.py, line 18
settings = {
    "reps": 6,  # Default, but typically overridden to 16
    ...
}

# ML/circuits.py, line 86
processing = variationalLayers(nQubits, vqcConf, maxLayers=vqcConf.reps)
#     reps=16
```

How to Set:

```
# Via command line
python runRegression.py '{"reps": 16}'

# Or in runRegression.py directly
settings = {"reps": 16, ...}
```

Where Used:

- `ML/circuits.py:parametrizedCircuit()` → Creates 16 variational layers
- Each layer contains: encoding gates (rx, rz) + entangling gates (cx)

5 Gates Configuration

5.1 Where Defined:

- **Primary**: `runRegression.py`, lines 16, 20-21
- **Processed in**: `ML/circuits.py`, lines 24-36

5.2 Encoding Gates: `["rx", "rz"]`

```
# runRegression.py, line 16
settings = {
    "encoding": ["rx", "rz"],  #     Encoding gates
    ...
}
```

Used in: `ML/circuits.py:encodingCircuit()` (line 39)

- Creates rotation gates: `qc.rx()` and `qc.rz()` for each qubit

5.3 Entangling Gates: `"cx"` (CNOT)

```
# runRegression.py, line 21
settings = {
    "entangle": "cx",  #     Entangling gate type
    ...
}
```

Used in: `ML/circuits.py:variationalLayers()` (lines 115-118)

```
# Circular entanglement pattern
qc.cx(nQubits - 1, 0)  # Last to first
for qubit in range(nQubits - 1):
    qc.cx(qubit, qubit + 1)  # Circular chain
```

5.4 Entanglement Type: `"circular"`

```
# runRegression.py, line 20
settings = {
    "entangleType": "circular",  #     Entanglement pattern
    ...
}
```

5.5 Measurement Basis: `"yz"`

```
# runRegression.py, line 19
settings = {
    "calc": "yz",  #     Measurement basis (Y and Z Pauli measurements)
    ...
}
```

Processed in: `ML/circuits.py:VQCsettings.__init__()` (line 31)

```
self.calcGates = stringToCircuitList(settings["calc"])  # ["y", "z"]
```

6 Training: 8000 Optimization Episodes

6.1 Where Defined:

- **Primary**: `runRegression.py`, line 23 (default: 1, but overridden to 8000)
- **Used in**: `ML/GradientQML.py`, line 50

```
# runRegression.py, line 23
settings = {
    "numEpisodes": 1,  # Default, but typically overridden to 8000
    ...
}

# ML/GradientQML.py, line 50
for episode in range(self.numEpisodes):  #     8000 episodes
    ...
```

How to Set:

```
# Via command line
python runRegression.py '{"numEpisodes": 8000}'

# Or in runRegression.py directly
settings = {"numEpisodes": 8000, ...}
```

Where Used:

- `ML/GradientQML.py:run()` → Main training loop (lines 50-75)
- Each episode: forward pass, loss computation, backward pass, optimization step

7 Optimizer: Adam with Learning Rate Decay

7.1 Where Defined:

- **Primary**: `runRegression.py`, lines 24-25
- **Configured in**: `ML/GradientQML.py`, lines 22-38

7.2 Optimizer Type: `"Adam"`

```
# runRegression.py, line 24
settings = {
    "optimizer": "Adam",  #     Optimizer type
    ...
}
```

Configured in: `ML/GradientQML.py`, lines 28-29

```
if self.settings["optimizer"].lower() == "adam":
    self.optimizer = Adam(self.agent.model.parameters(), lr=lr, amsgrad=True)
```

7.3 Learning Rate Decay: `[0.01, 100, 0.9]`

```
# runRegression.py, line 25
settings = {
    "lr": [0.01, 100, 0.9],  #     [initial_lr, step_size, gamma]
    ...
}
```

Interpretation:

- `0.01`: Initial learning rate
- `100`: Decay every 100 episodes
- `0.9`: Multiply LR by 0.9 at each decay step

Configured in: `ML/GradientQML.py`, lines 35-36

```
if isinstance(lrSettings, list):
    self.scheduler = StepLR(self.optimizer, step_size=lrSettings[1], gamma=lrSettings
    # StepLR(optimizer, step_size=100, gamma=0.9)
```

Applied in: `ML/GradientQML.py`, line 66

```
self.scheduler.step()  # Decay learning rate every episode
```

8 Complete Settings Dictionary Example

For the experimental setup you described, the settings would be:

```
# runRegression.py
settings = {
    "features": "simple",
    "encoding": ["rx", "rz"],        # Encoding gates
    "reuploading": False,
    "reps": 16,                      #     16 VQC layers
    "calc": "yz",                    # Measurement basis
    "entangleType": "circular",      # Entanglement pattern
```

```
    "entangle": "cx",                 # Entangling gate
    "reward": "rational",
    "numEpisodes": 8000,              #     8000 episodes
    "optimizer": "Adam",              #     Adam optimizer
    "lr": [0.01, 100, 0.9],          #     LR decay: start=0.01, step=100, gamma=0.9
    "prefix": "Paper",
    "noise": 0,
    "seed": 42,
    "batchsize": 1,
    "loss": "linear",                 # or "rational", "threshold", etc.
    "data": "stats/statsCards6",      #     Up to 6 tables
    "valueType": "rows",              # or "rowFactor" for QCardCorr
    "numFeatures": 6                  #     6 qubits (can be auto-detected)
}
```

9 Summary: File Locations

Parameter	Location	Line(s)
Up to 6 tables	`cardEnv.py`	50 (auto from data)
6 qubits	`ML/models.py`	39-41 (calculated)
16 VQC layers	`runRegression.py`	18 (`reps`)
Gates (rx, rz, cx)	`runRegression.py`	16, 20-21
8000 episodes	`runRegression.py`	23 (`numEpisodes`)
Adam optimizer	`runRegression.py`	24 (`optimizer`)
LR decay	`runRegression.py`	25 (`lr`)
LR scheduler	`ML/GradientQML.py`	35-36 (configured)

10 How to Run with These Settings

```
# Command line (recommended for experiments)
python runRegression.py '{
␣␣␣␣"reps":␣16,
␣␣␣␣"numEpisodes":␣8000,
␣␣␣␣"optimizer":␣"Adam",
␣␣␣␣"lr":␣[0.01,␣100,␣0.9],
␣␣␣␣"data":␣"stats/statsCards6",
␣␣␣␣"numFeatures":␣6,
␣␣␣␣"prefix":␣"Paper"
}'
```

Or modify `runRegression.py` directly (lines 14-36).

11 Where Settings Are Saved

After running, the complete settings are saved to:

- `results/settings/*.conf` - Human-readable settings file
- **Filename contains all settings** - See `ML/QML.py`, lines 34-36

  Example filename:

`Paper_simple_['rx',-'rz']_False_16_yz_circular_cx_rational_8000_Adam_[0.01,-100,-0.9]_Test_4`

  This filename encodes all settings, making it easy to identify the experimental configuration.