Documentation: Running the QAC-GAN Training and Testing Framework

1 Overview

This document provides step-by-step instructions for executing the Quantum AIO–ChameleonGAN (QAC-GAN) training and testing framework. The implementation reproduces the objective function, optimization procedure, and anomaly decision logic described in the study:

> *"An Evolution Quantum Camouflage Detection Algorithm for Polymorphic Cyber Attacks"*

The code implements:

- Quantum Variational Adversarial Learning (QVAL)
- Angle-of-Incidence Optimization (AIO)
- Temporal Evolution Tracking (TET)

and corresponds exactly to Equations (1)–(24) and Algorithms 1–4 in the manuscript.

2 System Requirements

2.1 Hardware (Recommended)

Component	Specification
CPU	Intel i7 / i9 or equivalent
RAM	$\geq$ 16 GB
GPU	Optional (NVIDIA CUDA-enabled recommended)
OS	Ubuntu 20.04+ / Windows 10+

Table 1: Recommended hardware requirements

Note: GPU acceleration improves training speed but is not mandatory.

2.2 Software Requirements

Install the following environment:

- Python $\geq$ 3.9
- TensorFlow $\geq$ 2.16
- NumPy $\geq$ 1.23
- SciPy $\geq$ 1.10
- scikit-learn $\geq$ 1.3

Installation Command

```
pip install tensorflow numpy scipy scikit-learn
```

Verification

```
python -c "import tensorflow as tf; print(tf.__version__)"
```

3 Code Structure

The implementation is modular and organized as follows:

Module	Description
QVALGenerator	Quantum-inspired generator
Discriminator	Geometry-aware discriminator
incidence_angle()	AIO computation
temporal_evolution_score()	TET computation
qac_gan_loss()	Composite loss (Eq. 11)
train_qac_gan()	Training loop
detect_anomaly()	Testing and classification
ablation_experiment()	Statistical validation

Table 2: QAC-GAN code structure

4 Dataset Preparation

4.1 Input Format

The training and testing functions expect:

$$X \in \mathbb{R}^{N \times d}$$

where:

- N is the number of samples
- d is the feature dimension

Each row represents a network traffic feature vector.

4.2 Preprocessing (Required)

Before training:

- Handle missing values (median or mode imputation)
- Normalize features using Min–Max scaling
- Split the dataset into 70% training and 30% testing

Example

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X_raw)

X_train = X_scaled[:int(0.7 * len(X_scaled))]
X_test  = X_scaled[int(0.7 * len(X_scaled)):]
```

5 Training the QAC-GAN Model

5.1 Basic Training Execution

```
G, D, mu_b = train_qac_gan(
    X_train,
    latent_dim=32,
    epochs=1000,
    lambda_aio=1.0,
    lambda_tet=1.0
)
```

Parameter	Meaning
latent_dim	Dimension of latent variable z
epochs	GAN training iterations
lambda_aio	Angular regularization weight
lambda_tet	Temporal drift penalty weight
mu_b	Learned benign baseline centroid

Table 3: Training parameters

5.2 Parameter Description

5.3 Expected Output

During training, the console will display:

```
Epoch 0   | Loss: ...
Epoch 100 | Loss: ...
Epoch 200 | Loss: ...
```

A monotonically stabilizing loss indicates correct convergence.

6 Testing and Anomaly Detection

6.1 Single-Sample Detection

```
result = detect_anomaly(
    x = X_test[0],
    D = D,
    mu_b = mu_b
)

print(result)
```

6.2 Output Interpretation

$$(\text{is\_anomaly}, \text{class\_label}, \text{score}, \theta_t, \text{TES})$$

6.3 Classification Rules (Eq. 24)

7 Batch Evaluation (Recommended)

```
results = [detect_anomaly(x, D, mu_b) for x in X_test]
```

Output	Meaning
is_anomaly	Boolean anomaly decision
class_label	Camouflage-aware classification
score	Discriminator confidence
θ_t	Incidence angle
TES	Temporal Evolution Score

Table 4: Detection output interpretation

Angle Range	Classification
0°–20°	Benign / Camouflaged
20°–25°	Stealth
$> 25°$	Mutating / Erratic

Table 5: Camouflage-aware classification rules

From these results, precision, recall, F1-score, and anomaly detection rate can be computed.

8 Ablation and Statistical Validation

8.1 Run Ablation Experiments

```
ablation_experiment(seeds=10)
```

8.2 Output

The function reports:

- Paired t-test
- Wilcoxon signed-rank test

These results reproduce Table 4 in the manuscript and confirm statistical significance.

9 Reproducibility Guidelines

To ensure exact reproducibility:

```
np.random.seed(42)
tf.random.set_seed(42)
```

Use identical dataset splits, random seeds, and training budgets.

10 Common Issues and Troubleshooting

Issue	Solution
Training diverges	Reduce learning rate
NaN loss	Normalize data, add epsilon
Slow convergence	Increase epochs
High false positives	Increase τ_θ threshold
GPU not detected	Verify CUDA installation

Table 6: Troubleshooting guide

11 Extending the Framework

This framework supports:

- Qiskit-based real quantum circuits
- CIC-UNSW-NB15 full dataset integration
- Online learning
- Reinforcement learning adaptation
- Hybrid MILP + GAN scheduling

12 Citation

If using this code, cite:

> Fondo, E., & Tole, K. (2026). *An Evolution Quantum Camouflage Detection Algorithm for Polymorphic Cyber Attacks.*