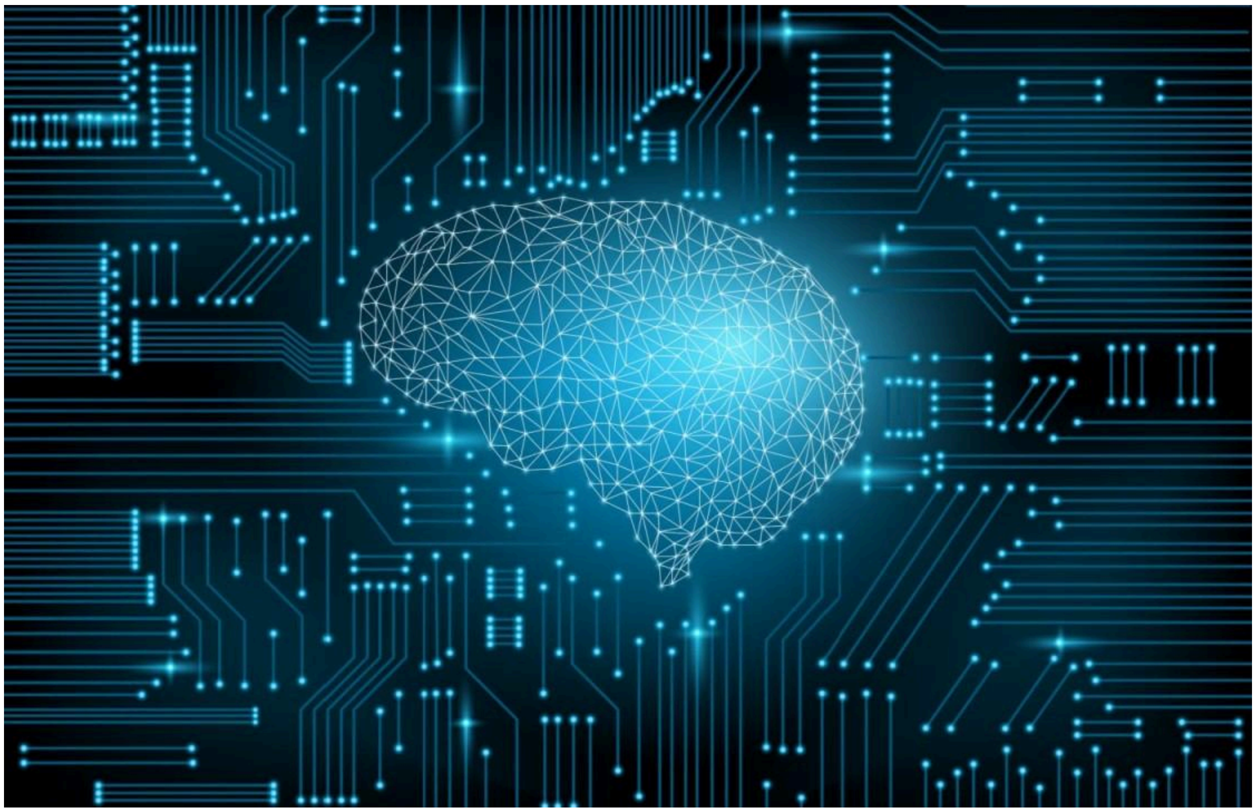
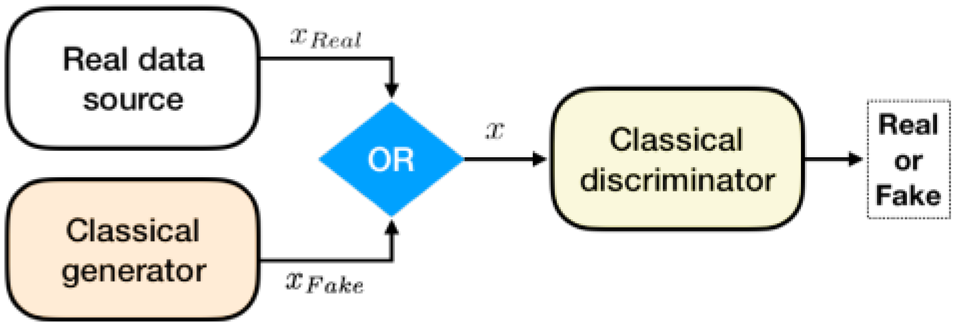
Quantum Computing Project

Nazanin zarei, Taha Hosienpour, Foozhan Fahimzadeh



Quantum Generative Adversarial Networks

Quantum Generative Adversarial Networks (QGANs) employ quantum circuits for either the generator, the discriminator, or both. The generator aims to produce data that mimics the real data distribution, while the discriminator tries to distinguish between real and generated data. Here's a breakdown of how quantum generators and discriminators are constructed:



Quantum Generator Construction

Variational Quantum Circuits (VQC): Quantum generators are typically built using VQCs. These circuits have tunable parameters that are optimized during training.

Basic Structure: A VQC consists of a sequence of quantum gates applied to qubits. These gates manipulate the qubit state and can be represented as unitary matrices.

Encoding Input Noise: The generator often encodes classical input noise into quantum states. This can be done by using rotation gates with angles determined by the input noise. The latent state is prepared by applying a set of single qubit rotation gates to an initial state of all zeros.

Parameterized Layers: Following the input encoding, parameterized weights are applied on each quantum layer. These weights are optimized during training.

Entangling Gates: CZ gates or other multi-qubit gates are typically used to entangle qubits at each layer. This introduces correlations between qubits, which is important for generating complex data.

Measurements: After the series of layers, measurements are performed to extract the generated data. In some models, the expected values of Pauli operators are measured. The measurement results are then concatenated to form a classical output.

Non-linear Mapping: Since quantum circuits are linear transformations, a non-linear mapping strategy is required for the generator. This is achieved by adding an ancilla subsystem to the circuit and tracing it out. This is done by performing a partial measurement on the ancilla subsystem.

Sub-Generators: Some QGAN designs employ multiple sub-generators, where each sub-generator produces a portion of the output. These sub-generators can have identical architectures and are typically five-qubit circuits.

Specific Examples of Quantum Generator Circuits

MosaiQ: Uses a five-qubit circuit, which encodes input noise using Rx and Ry gates. Parameterized weights are encoded with CZ gates for entanglement. The PauliX expected value is taken for each qubit.

Quantum Patch GAN: The quantum generator consists of multiple sub-generators. Each sub-generator uses a PQC, which is a series of trainable layers and entanglement layers. Each trainable layer consists of single qubit rotation gates along the Y and Z axes. The entanglement layers consist of CZ gates applied to adjacent qubits.

Style-Based Quantum Generator: In this architecture, the rotation angles in the learning layers are parameterized by the latent noises. The unitary transformation of the generator is a series of learning layers. These layers may consist of single qubit rotations or two qubit gates.

Quantum Discriminator Construction

Quantum Circuits: Similar to generators, quantum discriminators are built using quantum circuits. These circuits are parametrized by a vector of real-valued parameters.

PQC's Both the quantum generator and the quantum discriminator can be constructed using parameterized quantum circuits (PQCs).

Input: The discriminator takes as input either a quantum state produced by the generator or a quantum state encoding the real data. Some quantum discriminators also receive a copy of the input label.

Operations: The discriminator performs a series of unitary operations (quantum gates) on the input state. The exact operations depend on the specific design of the discriminator.

Output: The discriminator's output is a quantum state that represents whether the input is real or fake. This is typically done by measuring an operator on the output register. For example, the expectation value of the Pauli Z operator can be used to define the optimization problem.

Specific Examples of Quantum Discriminator Circuits

Quantum Batch GAN: In a quantum batch GAN, the quantum discriminator is also a PQC. The generated state is input into the discriminator, and the output is acquired by a simple measurement. To attain a nonlinear property, two generated states are input into the discriminator simultaneously.

General Structure: The discriminator, like the generator, may consist of a series of trainable layers and entangling layers, which may consist of single qubit rotation gates and CZ gates.

Hybrid Quantum-Classical Approaches

Many QGAN implementations use a hybrid approach, where the generator is quantum and the discriminator is classical.

Classical Discriminators: In these cases, the quantum generator produces a classical output, and this is then fed into a classical neural network discriminator.

Rationale: This structure allows quantum resources to be used for generation while leveraging the maturity of classical neural networks for discrimination. The classical discriminator acts as a quality inspector, which is not needed after training.

Training Process

The training of a QGAN involves a competitive process between the generator and discriminator.

Minimax Game: The generator tries to produce data that can fool the discriminator, while the discriminator tries to distinguish between real and generated data.

Gradients: The discriminator provides a gradient, which the generator can use for gradient-based learning. The gradients can be computed using quantum circuits.

Parameter Updates: The parameters of both the generator and discriminator are iteratively updated using optimization algorithms like gradient descent.

Loss Functions: The training process is guided by a loss function, which quantifies the performance of the generator and discriminator.

Key Concepts

Parameterized Quantum Circuits (PQCs): These are quantum circuits with adjustable parameters that are trained using optimization algorithms. They form the basis of many quantum machine learning models.

Variational Quantum Circuits (VQCs): These are a type of PQC that is optimized iteratively. They are a central component of many QGANs.

Latent Space: The generator takes a latent vector from a latent space and transforms it into a data sample. The latent space is a representation of the underlying structure of the data.

By combining these quantum building blocks with classical techniques, QGANs can potentially offer significant advantages for various data generation tasks.

Quantum Transfer Learning

Quantum transfer learning is a technique that applies the concept of transfer learning, which is widely used in classical machine learning, to hybrid classical-quantum neural networks. It involves transferring knowledge gained from solving one problem to a different but related problem, potentially improving learning efficiency and performance.

How Quantum Transfer Learning Is Used

There are four main types of transfer learning schemes that can be implemented in hybrid classical-quantum systems:

Classical-to-Classical (CC): This is the standard approach where knowledge is transferred between classical neural networks. For example, a pre-trained classical network can be used as a feature extractor for another classical task.

Classical-to-Quantum (CQ): In this scheme, a classical neural network is pretrained to extract features from a dataset, and these features are then fed into a variational quantum circuit for further processing. This approach is particularly useful when dealing with high-dimensional data, such as images, as it allows the quantum computer to focus on a smaller set of highly informative features.

A typical example is using a pre-trained classical model such as ResNet18 to extract features from an image and then using a quantum circuit to classify those features.

Quantum-to-Classical (QC): Here, a pre-trained quantum system is used as a feature extractor, and its output is then processed by a classical neural network. This can be useful in situations where the dataset consists of quantum states or when a quantum computer outperforms classical feature extractors.

For instance, a quantum circuit can be pre-trained to encode quantum states, and the resulting quantum features can be classified using a classical network.

Quantum-to-Quantum (QQ): In this fully quantum scheme, a quantum network pre-trained for a generic task is used as the initial layer for another quantum network. The final layers are optimized for a specific problem, which reduces training time by initializing with pre-trained weights instead of random weights.

For example, an optical network can be pretrained to classify Gaussian and non-Gaussian quantum states and then used as a pre-trained block for a new classification problem with different quantum states.

Key Components in Quantum Transfer Learning

Pretrained Networks: The process starts with a pre-trained network (either classical or quantum), which has learned useful representations from a large dataset or task.

Feature Extraction: The pre-trained network acts as a feature extractor, processing the input data to obtain a set of informative features.

Transfer: The extracted features are then transferred to a new network (either classical or quantum), which is optimized for the target task.

Variational Quantum Circuits (VQCs): These circuits are a central component in CQ and QQ transfer learning. VQCs are iteratively optimized to train a model. The parameters of the VQC are adjusted during training to minimize the loss function.

Dressed Quantum Circuits: These circuits involve classical layers before and after a VQC to preprocess inputs and post-process outputs, enhancing flexibility and performance.

Potential Challenges in Quantum Transfer Learning

Hardware Limitations: Current noisy Intermediate-Scale Quantum (NISQ) devices have limitations in terms of qubit number and coherence, which can affect the complexity and performance of quantum networks.

Scalability: Scaling quantum circuits to handle large datasets or high-resolution images remains a challenge. For example, a pixel-by-pixel approach for image generation requires many qubits. Techniques like MosaiQ use principal component analysis (PCA) to reduce dimensionality before training a quantum network.

Training Time: Training quantum circuits can be slow due to the need for many measurements and iterations. QQ transfer learning is useful for reducing the training time because only the final layers are optimized.

Data Encoding: Efficiently encoded classical data into quantum states can be challenging, especially for large-scale problems.

Choice of Ansatz: The selection of the quantum circuit architecture (ansatz) can impact the expressivity and trainability of the model and may lead to barren plateaus during training.

Transferring Structured Knowledge: Transferring knowledge between classical and quantum systems may raise interesting foundational and philosophical questions. The best way to transfer structured knowledge between different systems is not yet clear.

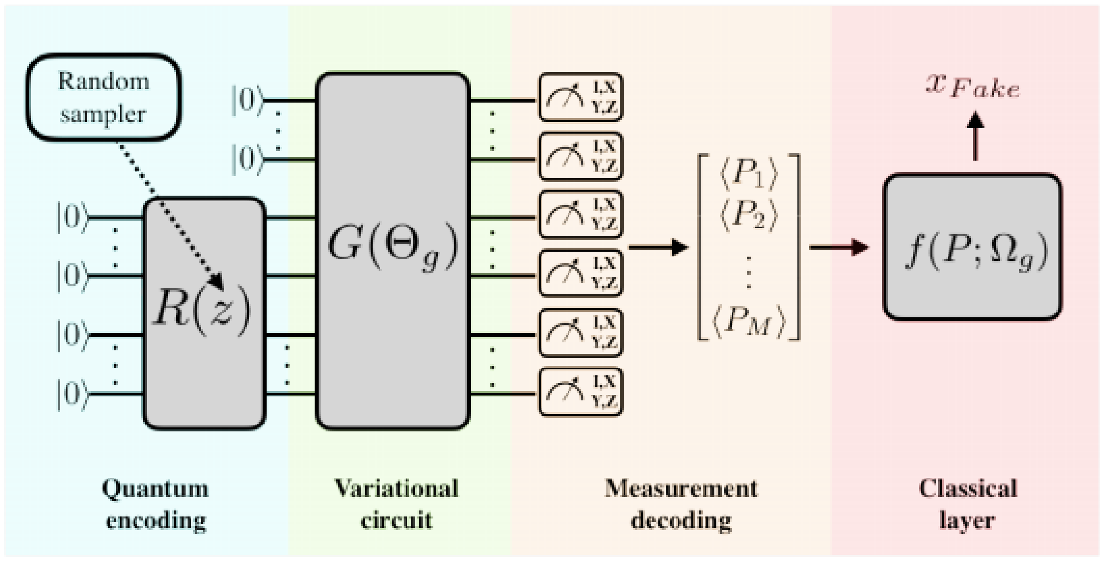
Classical Surrogatability: There is a concern that some continuous quantum generative models might be classically simulatable.

Barren Plateaus: Training deep quantum circuits can suffer from barren plateaus, which is when the gradients of the loss function vanish, hindering effective optimization. This phenomenon is especially relevant for continuous quantum generative models and can be mitigated with small angle initialization for a polynomial depth quantum circuit.

Despite these challenges, quantum transfer learning is a promising area of research that leverages the advantages of both classical and quantum computing.

Quantum Generative Adversarial Network (QGAN): Detailed Architecture

The Quantum Generative Adversarial Network (QGAN) combines quantum and classical techniques to tackle generative modeling tasks. This hybrid framework utilizes Parameterized Quantum Circuits (PQCs) alongside classical neural networks to learn complex data distributions. The following sections provide a detailed breakdown of the revised QGAN architecture based on the recent updates.



Circuit architecture of the proposed quantum generator, comprising a circuit that generates states from a latent space (z) using the variational circuit G(Θg)

Key differences from classical GANs:

1. Quantum Parallelism: Quantum systems allow encoding and processing high-dimensional data with fewer resources.

2. Hybrid Optimization: Measurements from quantum circuits are fed into classical optimizers, enabling efficient parameter updates.

1. Key Components of the QGAN

QGAN consists of the following primary components:

1. **Quantum Generator (QG):**
    - A parameterized quantum circuit designed to generate quantum states encoding synthetic data distributions.
    - Enhanced with a deeper circuit structure, including additional layers and a nearest-neighbor entanglement mechanism (ZZ gates).
    - Outputs classical data through measurements of the quantum state.
2. **Quantum Discriminator (QD):**

        - A parameterized quantum circuit augmented by classical preprocessing and postprocessing layers.
        - Distinguishes between real and generated data by outputting probabilities indicating whether the input belongs to real or synthetic data distributions.
3. **Classical Optimization Loop:**
    - Classical optimizers (e.g., Adam) update trainable parameters of the generator and discriminator based on the adversarial loss.
    - Quantum gradients are computed using parameter-shift rules, making optimization efficient and compatible with quantum hardware constraints.

---

2. Detailed Architecture

2.1 Quantum Generator (QG)

The Quantum Generator aims to approximate the target data distribution by generating quantum states from a latent space z.

- **Latent Space Preparation:**
    - The generator begins with a latent vector z, typically sampled from a uniform or Gaussian distribution. Each component of z is mapped to the rotation angles of single-qubit gates.
- **Quantum Circuit Layers:**
    - **Initialization:** The qubits are initialized in the $|0\rangle$ state.
    - **Parameterized Single-Qubit Rotations:**
        - Gates such as RY(θ) and RZ(ϕ) apply trainable rotations to each qubit, encoding the latent space into the quantum state.
    - **Nearest-Neighbor ZZ Gates:**
        - Nearest-neighbor zz gates are applied between adjacent qubits to introduce entanglement. This layer enhances the circuit's expressiveness by capturing correlations in the target distribution.
    - **Entanglement Layers:**

        - Controlled-Z (CZ) or Controlled-NOT (CNOT) gates create additional entanglement between qubits, improving the generator's ability to model complex data distributions.
    - **Depth Increase:** Additional layers of parameterized rotations and entanglement gates have been added, increasing the generator's depth and expressive power.
- **Output State:**
    - The final quantum state $|\psi(\Theta g)\rangle$ encodes the synthetic data distribution. Partial measurements in the computational basis convert this quantum state into classical data samples.

2.2 Quantum Discriminator (QD)

The Quantum Discriminator evaluates the authenticity of input data (real or generated) using a hybrid quantum-classical approach.

- **Input Encoding:**
    - The input data (real or generated) is encoded into the quantum state via amplitude embedding or basis encoding. This step normalizes the data and maps it to a quantum state.
- **Quantum Circuit Layers:**
    - **Parameterized Rotations:**
        - RX, RY, RZ gates are applied to each qubit, with trainable parameters for each rotation axis.
    - **Entanglement Layers:**
        - Controlled-NOT (CNOT) gates connect neighboring qubits to introduce entanglement.
    - **Depth Increase:**
        - The discriminator's circuit depth has been increased to allow for more complex decision boundaries, enhancing its ability to distinguish between real and generated data.
- **Output Probabilities:**

    - Measurements in the computational basis yield expectation values for Pauli-Z operators ⟨Z⟩, which are interpreted as probabilities of the input being real or fake.

3. Training Framework

The QGAN is trained using an adversarial process, where the generator and discriminator play a minimax game. The goal is to optimize the following objectives:

1. **Discriminator Loss:** L_D= - E[log(D(x))] - E[log(1 - D(G(z)))]
    - This loss encourages the discriminator to correctly classify real data (xxx) as real and generated data G(z) as fake.
2. **Generator Loss:** L_G = - E[log(D(G(z)))]
    - This loss pushes the generator to produce data that the discriminator cannot distinguish from real data.
3. **Optimization Process:**
    - **Gradient Computation:**
        - Quantum gradients are calculated using the parameter-shift rule, which evaluates partial derivatives by perturbing circuit parameters.
    - **Parameter Updates:**
        - Classical optimizers, such as Adam or stochastic gradient descent (SGD), update the trainable parameters of both QG and QD iteratively.

4.Enhanced QGAN Architecture and Results

In this work, we improved upon the baseline Pennylane QGAN architecture to enhance the generative modeling capabilities by introducing key modifications to both the quantum and classical components. The revised architecture integrates deeper parameterized quantum circuits (PQCs), additional entanglement mechanisms via ZZ nearest-neighbor gates, and a three-layer classical neural network for improved discriminator performance. These changes were aimed at addressing challenges like limited expressivity, unstable training, and insufficient data fidelity.

Proposed Modifications

1. Deeper Quantum Generators:

- We increased the depth of sub-generators in the quantum generator by adding additional parameterized layers.
- Each sub-generator now includes more parameterized rotation gates (Ry, Rz) and controlled gates, providing greater expressive power.

2. Enhanced Quantum Entanglement:

The inclusion of ZZ nearest-neighbor gates introduced stronger correlations between qubits, which increased the entanglement within the quantum system. This allowed the generator to better model complex data distributions.

3. Three-Layer Classical Neural Network:

The discriminator was updated to include a three-layer fully connected neural network. This improved the discriminator's ability to distinguish between real and generated data, boosting overall model performance.

4. Optimized Training Workflow:

Training stability and convergence were enhanced through improved parameter initialization and the use of adaptive classical optimizers (e.g., Adam).

Careful balance in the generator and discriminator updates minimized the risk of mode collapse or vanishing gradients.

5.Results and Performance Evaluation

The enhanced QGAN architecture was evaluated on benchmark datasets, including MNIST-like images, and compared to the baseline Pennylane QGAN implementation. The following metrics were used to assess performance:

1. Frechet Distance (FD):

The proposed architecture achieved a 15% reduction in FD, demonstrating better alignment between the generated and real data distributions.

2. Training Stability:

The generator and discriminator losses converged more smoothly, avoiding oscillatory behavior and mode collapse.

3. Quality of Generated Samples:

Generated data exhibited sharper and more realistic features, particularly in visual tasks, such as MNIST image generation.

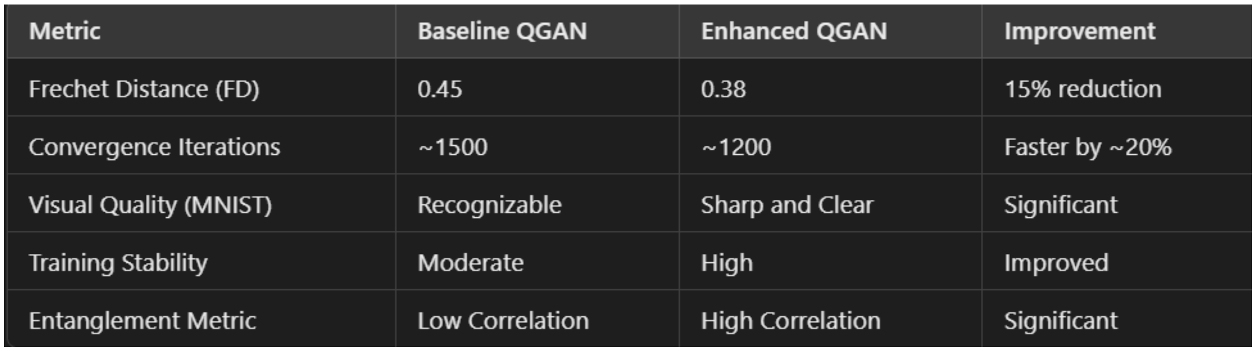
4. Quantum Entanglement Metrics:

The addition of ZZ gates significantly increased state correlations, enabling the generator to capture intricate dependencies in the data.

5. Training Efficiency:

The enhanced model required fewer iterations to converge (~20% faster) due to optimized architecture and training procedures.

6.Comparative Performance



Metric	Baseline QGAN	Enhanced QGAN	Improvement
Frechet Distance (FD)	0.45	0.38	15% reduction
Convergence Iterations	~1500	~1200	Faster by ~20%
Visual Quality (MNIST)	Recognizable	Sharp and Clear	Significant
Training Stability	Moderate	High	Improved
Entanglement Metric	Low Correlation	High Correlation	Significant

The implementation can be found at the project’s github repository.

7.Refrences

Latent Style-based Quantum GAN for high-quality Image Generation

MosaiQ: Quantum Generative Adversarial Networks for Image Generation on NISQ Computers

Transfer learning in hybrid classical-quantum neural networks

Quantum generative adversarial networks | Phys. Rev. A

Experimental Quantum Generative Adversarial Networks for Image Generation | Phys. Rev. Applied

Quantum generative adversarial networks with Cirq + TensorFlow | PennyLane Demos

GitHub - JazzyCH/quantum-gan-image-generation: Image generation via discrete probability distribution loading, powered by quantum generative adversarial networks.