FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Incremental Learning of Quantum Generative Adversarial Network
Student:	Bc. Artem Kandaurov
Supervisor:	Ing. Ivo Petr, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2022/2023

Instructions

With growing computational capabilities the field of machine learning has seen rapid development, and finds applications in many human activities. At the same time enormous effort is devoted to research quantum computers and algorithms that can potentially outperform their classical counterparts.

The goal of the thesis is to investigate possible application of quantum technologies in machine learning. Namely, the student will:
a) get acquainted with principles of quantum algorithms and generative adversarial networks (GANs)
b) investigate benefits following from application of quantum technologies in machine learning with focus on quantum generative adversarial networks (qGANs)
c) develop method that allows incremental learning of qGANs and implement it using Qiskit - open source software development kit for working with quantum computers
d) test the concept and implementation of qGAN with incremental learning on a specific problem of probability distribution learning

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 17 February 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Incremental Learning of Quantum Generative Adversarial Network

Bc. Artem Kandaurov

Department of Computer Science
Supervisor: Ing. Ivo Petr, Ph.D.

May 6, 2021

Acknowledgements

I am incredibly grateful to my supervisor Ing. Ivo Petr, Ph.D. for the great help with this thesis. This work would not exist without you, thank you very much! Also, I would like to thank my wife Ekaterina for her care and support.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 6, 2021 …………………

Citation of this thesis

Kandaurov, Artem. *Incremental Learning of Quantum Generative Adversarial Network*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstract

Machine learning field has shown incredible impact on many kinds of optimization problems. Recently the power of machine learning was applied to speed up the quantum states preparation. Although approximation with quantum generative adversarial networks is one of the fastest ways to prepare a generic quantum state, training time for such models is still significant and can easily impair quantum advantage. This thesis explores incremental learning of quantum generative adversarial networks for the quantum states preparation problem and introduces learning use cases reducing the training time.

Keywords Quantum generative adversarial network, Incremental learning, Quantum state preparation, Quantum machine learning, QGAN

Abstrakt

Obor strojového učení ukázal neuvěřitelný dopad na mnoho druhů optimalizačních problémů. Nedávno byla síla strojového učení použita k zrychlení přípravy kvantových stavů. Navzdory skutečnosti, že aproximace stavu pomocí kvantové generativní soupeřící sítě je jeden z nejrychlejších způsobů přípravy generického kvantového stavu, doba trénovaní pro takové modely je významná a může snadno eliminovat výhody plynoucí z použití kvantového algoritmu. Tato práce zkoumá využití inkrementalní učení kvantové generativní soupeřící sítě pro problém nahrání kvantových stavů a ukazuje nové případy použití v nichž se zkracuje trénovaní modelu.

Klíčová slova Kvantová generativní soupeřící síť, Inkrementální učení, Nahrání kvantového stavu, Kvantové strojové učení

Contents

List of Figures

Introduction

Quantum computing is nothing more than quantum mechanics effects used for developing algorithms, but such techniques applied to algorithmic problems have shown a fabulous outcome. Quantum algorithms already outperformed their classical counterparts for some tasks and even demonstrated exponential speed up [1]. Present quantum computers are still far from solving real problems, but the fairy tale will come true sooner or later, and quantum computing will become as common as classical computing. However, there are theoretical and practical challenges to be solved first.

One of the challenges is the efficient loading of classical data that we want to process on a quantum computer into quantum states. Quantum algorithms obviously work only with quantum data, but the complexity of quantum state preparation can diminish or completely eliminate potential quantum advantage. Loading particular generic state in n qubits requires $\mathcal{O}(2^n)$ gates [2]. Fortunately, there is no need to load exact quantum states. For example, quantum amplitude estimation algorithm [3] or quantum algorithm for solving linear systems of equations [4] require only a fair approximation of a quantum state. Furthermore, quantum computers also produce errors, and loading exact quantum states under these conditions does not make sense.

Machine learning adapted to quantum computers can be used to prepare approximated quantum states. Quantum generative adversarial networks [5] are already being used for quantum states estimation [6] and loading probability distributions into quantum registers [7]. It is a very new and promising branch of quantum machine learning that significantly decreases the complexity of preparing generic states up to $\mathcal{O}(poly(n))$ gates [7].

However, some of these gates, named Pauli rotation gates, have variable parameters that should be trained. The training time is the main disadvantage of using quantum generative adversarial networks for quantum states loading [8]. This work explores incremental learning of quantum generative adversarial networks that could significantly reduce the training time in real-world scenarios.

Chapter 1

Basics of Quantum Computation

Quantum computing is a potent instrument that uses quantum physical features to perform computations. The most important quantum mechanical effects used in quantum computing are *superposition*, i.e., the ability of quantum states to be added together forming another valid quantum state, and *entanglement*, i.e., interaction of quantum particles making their properties perfectly correlated. These effects are crucial for quantum computing and help quantum algorithms outperform the best known classical algorithms. But first things first.

1.1 Quantum Information

Bit is a classical information unit. Quantum information differs from the classical one and operates with qubits, i.e., quantum bits. Qubit is defined as an element of two-dimensional Hilbert space $\mathcal{H}$ and can be expressed as linear combination of computational basis states that are denoted as $|0\rangle$ and $|1\rangle$. Here $|0\rangle$ and $|1\rangle$ are quantum states in the Dirac notation [9] that will always be measured to classical 0 and 1 respectively.

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.1}$$

Measurement is a distinguished operation in quantum computing because it changes the state of a qubit. After measurement, qubit collapses into a computational basis state with probability $|k|^2$, where $k \in \mathbb{C}$, called *amplitude*, is the coefficient in the linear combination of the computational basis state. For example, qubit $|\Psi\rangle$ from equation 1.1 will be measured as $|0\rangle$ with probability $|\alpha|^2$ or as $|1\rangle$ with probability $|\beta|^2$. Obviously, a sum of all probabilities should be equal to one.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1.2}$$

The qubit $|\Psi\rangle$ can be written in a vector notation as $[\alpha \;\; \beta]^T$. As the sum of squared absolute values of coefficients has to be 1, any valid qubit also can be represented as a point on a sphere, so-called *Bloch sphere* [10]. Equation 1.3 is the geometric notation of a qubit defined by parameters $\theta \in \mathbb{R}$ and $\varphi \in \mathbb{R}$.

$$|\Psi\rangle = cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}sin\frac{\theta}{2}\,|1\rangle \tag{1.3}$$

Figure 1.1 shows a visual representation of a Bloch sphere. Note that any point on a sphere can be defined by only two angle parameters.

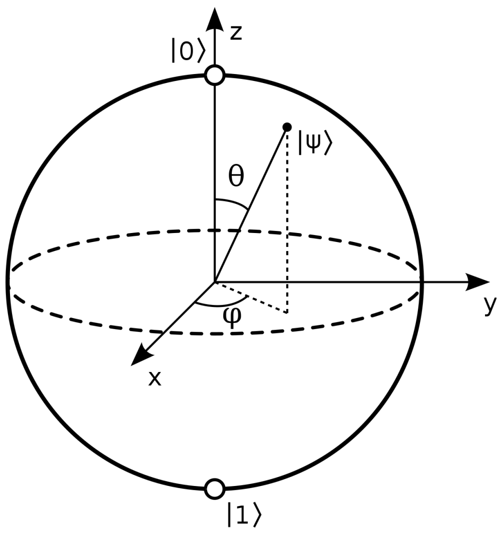

Figure 1.1: Bloch sphere representation of a qubit.

Qubits can be combined into a *quantum register* to form a quantum state, e.g., $|00\rangle$, also can be denoted as $|0^2\rangle$. More generally, quantum state defined by a quantum register consisting of n qubits is an element of tensor product $\mathcal{H} = \mathcal{H}_{n-1} \otimes ... \otimes \mathcal{H}_0$ and thus is specified by 2^n amplitudes [10].

The mathematical definition of a qubit allows it to be in a superposition of basis states, but does it exist in the real world? Physically qubit can be represented with quantum particles, e.g., photons polarization angle or electron levels in an atom. The field of quantum computer building is developing very quickly, and recently a quantum computer implemented via acoustic resonators was presented [11]. The physical structure of quantum computers is another fascinating topic, but it is out of the scope of this thesis.

1.2 Quantum Computation

Operation on a qubit or multiple qubits is called a *gate*. Due to postulates of quantum mechanics, every gate has to be unitary operator that can be represented as unitary matrix. There are many gates defined, e.g., Hadamard gate, Toffoli gate, or Pauli rotation gates. Actually, any unitary matrix specifies a valid quantum gate [10]. Thus there are infinitely many quantum gates, but the gates mentioned in this thesis are reviewed below.

1.2.1 Hadamard gate

The Hadamard gate (**H**) is a unary operation that is often used for superposition creating. The operation matrix is presented in equation 1.4.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.4}$$

Equations 1.5 and 1.6 refer to common quantum states denoted as $|+\rangle$ and $|-\rangle$ for simplicity. These states represent superposed quantum states with equal probabilities of measuring $|0\rangle$ or $|1\rangle$ basis state.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{1.5}$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.6}$$

The Hadamard gate applied on the quantum state $|0\rangle$ maps it to the $|+\rangle$ state and applied on the quantum state $|1\rangle$ maps it to the $|-\rangle$ state. These facts are demonstrated by equations 1.7 and 1.8. These are the most common examples, but quantum gates can be applied to any valid quantum state.

$$H\,|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |+\rangle \tag{1.7}$$

$$H\,|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |-\rangle \tag{1.8}$$

1.2.2 Pauli rotation gates

Pauli rotation gates (**RX**, **RY**, **RZ**) have an input parameter θ with rotation angle about x, y, or z axis according to the gate. Operations matrices are presented in equations 1.9, 1.10, and 1.11.

$$RX(\theta) = \begin{bmatrix} cos(\frac{\theta}{2}) & -i \cdot sin(\frac{\theta}{2}) \\ -i \cdot sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \tag{1.9}$$

$$RY(\theta) = \begin{bmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{bmatrix} \tag{1.10}$$

$$RZ(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{1.11}$$

Any quantum state can be prepared using Pauli rotation gates [10]. In fact, the Hadamard gate can be replaced with two rotation gates.

1.2.3 Controlled Pauli rotation gates

Controlled Pauli rotation gates (**CX**, **CY**, **CZ**) act on two qubits, where the first one acts as a control for applying a rotation operation on the second one. The operation is only performed if a control qubit is $|1\rangle$; otherwise, the gate has no effect. Controlled Pauli rotation gates can be parametrized the same way as the original Pauli rotation gates, but by default, they rotate a quantum state by π radians around the axis. Operations matrices are presented in equations 1.12, 1.13, and 1.14.

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.12}$$

$$CY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix} \tag{1.13}$$

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{1.14}$$

Controlled gates applied on two qubits can entangle them, making them perfectly correlated. For example, perfectly correlated *Bell state* [12] from equation 1.15 can be easily created with controlled Pauli-X gate applied on $|+\rangle$ control qubit and $|0\rangle$ target qubit.

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.15}$$

1.2.4 Quantum circuit

Quantum circuit is an ordered set of gates applied on quantum and classical registers. Most often, the resulting quantum state is measured at the end of quantum circuit. After measurement qubits are collapsed to $|0\rangle$ or $|1\rangle$ and result of the measurement, i.e., 0 for $|0\rangle$ and 1 for $|1\rangle$, is written into assigned classical register. Measurement is a non-unitary operation and cannot be reverted.

Elementary example of a quantum circuit is showed in figure 1.2. The circuit is read from left to right. Initial quantum state $|00\rangle$ is transformed into superposition with Hadamard gates mapping it to $H|0\rangle \otimes H|0\rangle$. Then the qubits are entangled with controlled Pauli-Z gate. Resulting state of the quantum circuit before measurement is $[0.5 \;\; 0.5 \;\; 0.5 \;\; -0.5]^T$. Note that every measurement of the state is equivalent to generating samples from a uniform probability distribution defined for values 0, 1, 2, and 3.

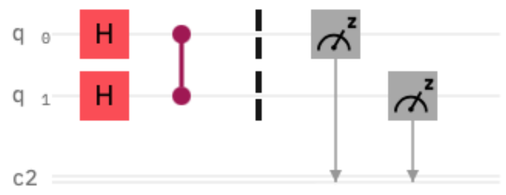

Figure 1.2: Quantum circuit representing uniform probability distribution with entangled qubits.

Vertical dotted lines, so-called *barriers*, are used for separation gates for the transpiler. The transpiler works with separated parts of quantum circuit individually without optimization. Also, barriers are used for visual separation of different parts of a quantum circuit.

Chapter 2

Quantum Machine Learning

Quantum machine learning is a very new and promising branch of the quantum computation field. Many new algorithms were developed recently, but mostly they are quantum adaptations of existing machine learning algorithms. Generally, quantum machine learning algorithms can be divided into quantum algorithms, hybrid classical-quantum algorithms, and classical algorithms applied to some quantum computation problems.

Pure quantum algorithms are based on existing quantum computation techniques to speed up calculations, e.g., quantum amplitude amplification or solving linear systems of equations algorithm. For example, pure quantum machine learning Bayesian inference algorithm [13] uses both techniques. Quantum generative adversarial network [5] is a good example of hybrid algorithms; it combines both quantum and classical parts. Classical machine learning, for example, can be used for optimizing settings for Bell non-locality [14].

Of course, the quantum machine learning field will expand with new algorithms and approaches. Quantum machine learning will probably take a more significant place in the quantum computation field than classical machine learning in classical computer science, primarily because of the quantum devices limitations.

2.1 Variational Circuits

One of the most important concepts in hybrid quantum-classical machine learning is *quantum variational circuit*, i.e., parametrized circuit. The circuit has a set of variable parameters that can be optimized during the learning process.

Quantum variational circuit consists of repeated parametrized rotation gates and entanglement blocks. Rotations are performed with **RX**, **RY**, or **RZ** gates. Thus optimizing parameters are rotation angles in radians. The rotation gates can be combined to add another degree of freedom, but it increases the circuit's optimization complexity. Qubits entanglement can be made with any controlled gates. Most often, **CZ** or **CX** gates are used.

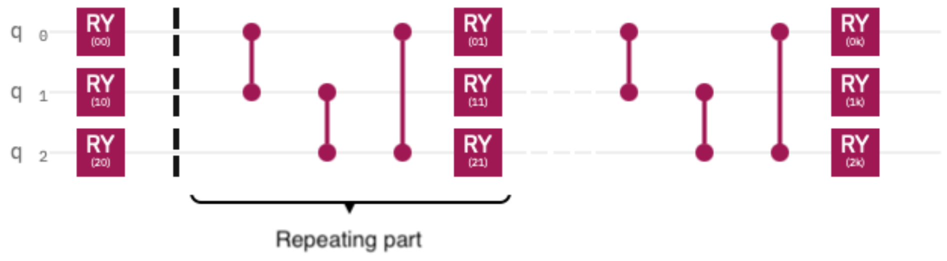

Figure 2.1: Quantum variational circuit based on Pauli-Y and CZ gates, number of qubits $n = 3$ and generic depth k.

Figure 2.1 presents a quantum variational circuit that is built with **RY** and **CZ** gates. The advantage of the circuit's architecture based on **RY** and **CZ** gates is that the circuit does not change initial quantum state with nullified parameters. Thus any loaded initial quantum state can be used as a starting point for further optimization.

Depth of the circuit refers to the number of entanglement and rotation blocks repetitions. Enlarged depth adds more variable parameters and, therefore, increases the circuit's optimization complexity. The complexity of the parametrized circuit should be tuned according to the optimization problem. In this sense, the quantum variational circuit's complexity is similar to artificial neural network architecture. With the parametrized circuit, the optimization problem is reduced to the approximation of rotation angles. The parameters can be trained with any classical optimization algorithm, for example, ADAM [15].

2.2 Generative Adversarial Network

Generative adversarial network [16] is a machine learning model. The main idea behind generative adversarial networks is training two neural networks contesting with each other. The first one, so-called *generator*, captures data distribution, and the second one, so-called *discriminator*, estimates the probability that a sample came from the training data rather than the generator. Both neural networks are parametrized with real vectors: the generator's parameters θ_g and the discriminator's parameters θ_d.

The training process corresponds to a minimax two-player game. The generator G_{θ_g} receives noise vector z taken from a prior distribution p_{prior} as input and produces data sample $G_{\theta_g}(z)$. The discriminator D_{θ_d} receives training data from a real distribution p_{real} mixed with samples from the generator and returns a probability $D_{\theta_d}(x)$ that passed sample is real. Then, the discriminator aims to maximize the probability of assigning the correct label to generated and real samples, and the generator simultaneously aims to minimize the discriminator's ability to detect generated samples. The model's training process is demonstrated in figure 2.2.

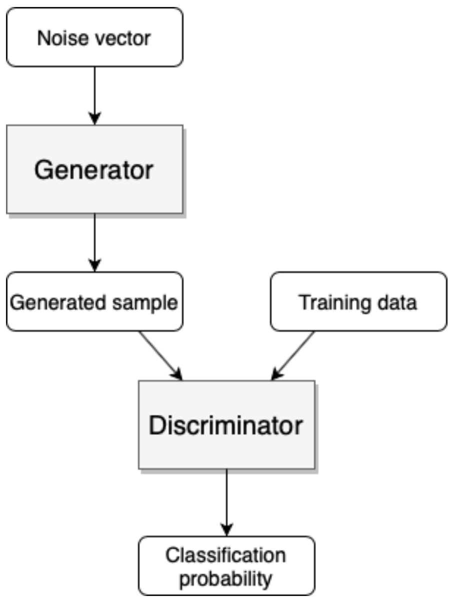

Figure 2.2: Generative adversarial network structure.

The discriminator, defined by parameters θ_d, and the generator, defined by parameters θ_g, are trained simultaneously with a gradient calculated from equation 2.1. In practice, ADAM optimizer [15] can be used for the updating steps.

$$\min_{\theta_g}(\max_{\theta_d}(\ \mathbb{E}_{x \sim p_{\text{real}}}[\log D_{\theta_d}(x)] + \mathbb{E}_{z \sim p_{\text{prior}}}[\log\Big(1 - D_{\theta_d}(G_{\theta_g}(z))\Big)])) \tag{2.1}$$

The minimax game target was defined for the original generative adversarial network. However, there are many modifications of loss functions. For example, non-saturating approach changes the generator's loss function to $-\log D_{\theta_d}(G_{\theta_g}(z))$ to avoid stagnation early in learning.

Generative adversarial networks showed incredible results and significantly affected the machine learning field. Many modifications of the original algorithm were developed, e.g., conditional generative adversarial networks [17], stacked generative adversarial networks [18], cycle-consistent generative adversarial networks [19] and others.

2.3 Quantum Generative Adversarial Network

Quantum generative adversarial network [5][6] is a very new algorithm. Quantum modification of the generative adversarial network was presented in different architectures: input data, generator, and discriminator can be defined as quantum or classical. This thesis aims to speed up quantum state, i.e., quantum generator, preparation with classical data, so the architecture with classical input data, quantum generator, and classical discriminator is considered.

The idea behind adversarial learning remains the same. The generator and the discriminator are trained simultaneously, contesting with each other. The main difference is that the generator is represented by a quantum variational circuit applied to some initial quantum state. The quantum variational circuit is defined by a vector of parameters θ_g that are angles in radians for the rotation gates. The parameters can be trained on the classical computer with any gradient-based optimization algorithm. The discriminator for the quantum version of the generative adversarial network can remain the same as for the original model.

$$|\Psi\rangle = \sum_{j=0}^{2^n-1} \sqrt{P(b_j)}\, |b_j\rangle \tag{2.2}$$

The main aim of generative adversarial networks is to train a generator to approximate the underlying process p_{real}. Generic quantum generator's output is defined in equation 2.2, where n is the number of qubits and $P(b_j)$ is the probability of measuring the basis state $|b_j\rangle$. Then, the learning target is to approximate output probabilities of each basis state.

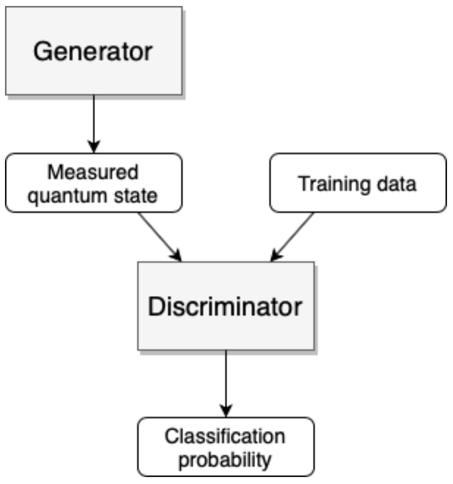

Figure 2.3: Quantum generative adversarial network structure.

To generate a data sample it is enough to measure the quantum generator's output $|\Psi\rangle$. Measured data g are purely classical and correspond to the trained probabilities.

Figure 2.3 demonstrates the model's learning process. In fact, quantum generative adversarial networks are more natural because a quantum generator does not require a randomized input. Stochasticity of the output is based on the quantum measurement uncertainty.

The gradient for updating the generator's and the discriminator's parameters is calculated basing on the loss functions.

$$L_G(\theta_g, \theta_d) = -\mathbb{E}[\log(D_{\theta_d}(g))] \tag{2.3}$$

$$L_D(\theta_g, \theta_d) = \mathbb{E}_{x \sim p_{\text{real}}}[\log D_{\theta_d}(x)] + \mathbb{E}[\log(1 - D_{\theta_d}(g))] \tag{2.4}$$

Equation 2.3 refers to the generator's non-saturating loss function and equation 2.4 to the discriminator's loss function. However, the loss functions can be changed the same way as for the classical generative adversarial network.

Chapter 3

Incremental Quantum Generative Adversarial Network

State of the art in theoretical quantum computing requires a creative approach for solving real problems with currently available quantum devices. The idea of quantum states approximation using quantum generative adversarial networks [20] significantly reduced the complexity of quantum state preparation. However, the time needed for quantum generative adversarial network training is still the issue.

The main aim of incremental learning exploration in this thesis is to speed up quantum state preparation, i.e., preparation of qubits in a quantum generator. Parameters of a generator can be updated via gradient descent with a classical discriminator on a classical computer [6]. That is why from here, only quantum generative adversarial network architecture with a quantum generator and a classical discriminator will be considered. But what is incremental learning?

Incremental learning is a training method in machine learning in which training data are passed to the model gradually when new samples become available. This approach considers data streams in contrast with a traditional assumption of complete data availability. Incremental learning can be helpful in changing environments or for lifelong learning and is more natural for real-time problems.

Quantum generative adversarial network, as a modification of classical generative adversarial network, inherently supports incremental learning. However, there are many concerning issues about its configuration, learning techniques, and possible advantages of incremental learning. Theoretically, incremental learning applied on a quantum generative adversarial network can lead to a significant reduction of the training time in real-world learning scenarios [7]. The contribution of this thesis proves this statement and also describes some compelling use cases.

3.1 Training Data

Any incremental learning scenario assumes the existence of a data stream with constantly arriving data [24]. This data stream represents some unknown underlying process, e.g., probability distribution. The quantum generative adversarial network aims to approximate the underlying process.

In the case of incremental learning, a target state should not be defined on the initialization but can depend on the online data source and vary over time. New data from the data stream can be used to specify the generator's parameters to meet the unaltered underlying process or modify them to be up-to-date with the time-varying underlying process. In other words, training strategy depends on the stationarity of the underlying process.

3.1.1 Stationary process

Stationary process is a process whose parameters do not change over time. A good example of a stationary process is samples generation from a probability distribution with constant parameters. Although the generated values may differ, the parameters of the underlying process remain the same. This property ensures the immutability of the process during samples generation and, therefore, during incremental learning. Stationary processes are simple and straightforward for understanding. However, there are some open questions in the context of incremental learning. Section 3.5.1 explains its specificities.

For applying a quantum algorithm in the real world, a quantum state with actual data may be needed, e.g., application of quantum amplitude estimation algorithm for the option pricing [8]. The time needed for collecting a big enough training dataset can prevail over speed-up obtained from quantum state approximation with a quantum generative adversarial network. The required size of the training dataset can be enormous, depending on the problem itself, data dimension, number of possible states, and the process complexity.

Incremental learning allows not to have all the data at the very beginning but update the initial dataset with new samples gradually. The larger the training dataset becomes, the more precisely the target state is defined. Instead of being idle while collecting data, quantum generative adversarial network could approximate an insufficiently accurate target state. In the overwhelming majority of cases, the quantum state will converge to the target reducing the relative entropy between actual and wanted quantum states.

3.1.2 Non-stationary process

Non-stationary process is, obviously, a process whose parameters are varying in time. Sometimes non-stationary process refers to the term *concept drift*. A straightforward example of a non-stationary process is samples generation from different probability distributions depending on time. Changing the underlying process parameters, the target state also changes. Changes can be gradual or abrupt; also, they can be only in a specific region of data space [24]. Section 3.5.2 describes the details about the non-stationary process learning.

Learning of such a process makes sense in the context of lifelong learning [24], i.e., continuous model updating with the new data received from a data stream. This technique requires training data prioritizing according to the time – the latest samples represent a more relevant data stream. Lifelong learning of a non-stationary process in the context of quantum generative adversarial networks allows having a relevant approximated quantum state at any time step during incremental learning. For some use cases, e.g., in the quantum finance field, it can completely reduce quantum state preparation complexity.

3.2 Generator

Incremental learning does not require any changes in the quantum generator's architecture. However, it is crucial to set up a quantum generator according to the problem. The quantum generator is a variational circuit applied on some prepared initial state, so the most significant parameters are depth and initial state. Section 2.1 provides more information about variational quantum circuits.

Underlying process complexity and data dimension, i.e., the number of qubits, are given by the problem, but other parameters heavily depend on them. Comparison of different depth and initial state parameters for quantum generative adversarial networks applied on a quantum state preparation problem can be found in paper [7].

Depth polynomially increases the number of optimizing parameters and should reflect the problem's complexity. More formally, depth k acting on n qubits of a variational circuit configured with one rotation gate per depth's layer has $n(k+1)$ parameters in total. Basically, the larger is data dimension and the more complex is underlying process, the greater the quantum generator's depth should be. On the other side, the quantum generator should not overpower the classical discriminator, so the depth constant should be chosen wisely.

The initial state of the quantum generator is also important, but primarily for stationary process learning. The choice very much defines the training time to reach acceptable relative entropy. Basically, the closer is initial state to the target distribution p_{real}, the faster acceptable result will be reached. However, in the context of non-stationary process learning, the initial state plays a role only at the beginning and practically does not affect the lifelong learning process.

For example, uniform probability distribution can be a good initial state. Uniform distribution has the advantage that it is effortless to prepare using a quantum circuit because Hadamard gates applied to all $|0\rangle$ qubits generate fair uniform distribution. Figure 3.1 presents the quantum generator circuit on three qubits with the uniform initial state followed by the parametrized two-layers circuit based on **RY** and **CZ** gates.

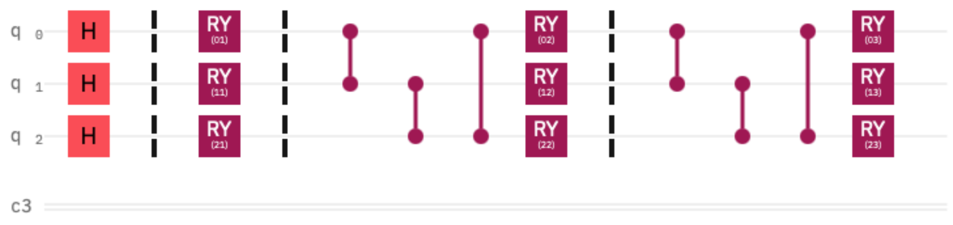

Figure 3.1: Quantum generator with uniform initial state, number of qubits $n = 3$ and depth $k = 2$.

3.3 Discriminator

The discriminator should distinguish between real data and generated samples and be able to improve its accuracy. Another essential requirement is providing a gradient which the generator can use for gradient-based learning. The discriminator used in the experiments is a classification neural network. The most significant tunable parameter in the context of incremental learning is the neural network's architecture.

Architecture of the neural network, meaning the number of hidden layers and nodes, is very similar to a quantum generator's depth. The complexity of a quantum generator and a classical discriminator should be balanced for effective adversarial learning. Moreover, both should meet the problem at hand. There is no optimal quantum generator's or classical discriminator's architecture because it is neither apriori clear what structure is the most suitable for a given problem [7].

3.4 Optimizer

The parameters of a quantum generator and a classical discriminator should be optimized during adversarial learning. This thesis uses AMSGRAD optimizer [25]. The optimizer has many configurable parameters and can be perfectly tuned according to the task, but the most interesting parameter for the incremental learning case is the learning rate parameter. The parameter defines the step size at each iteration while moving toward a minimum of a loss function [26].

Appendix A shows experiments results with variable learning rate for the stationary and non-stationary test cases. It follows from the experiments that the learning rate parameter can significantly affect the training process and, therefore, should be chosen carefully. Basically, lower learning rate causes slower converge to the target, but higher learning rate has a risk of fast diverging from the local optimum.

3.5 Learning

Implementation details of the incremental learning modification of a quantum generative adversarial network are described in section 5.1.1. The contribution of this thesis supports both stationary and non-stationary process learning. However, learning methods significantly differ depending on the underlying process type.

3.5.1 Stationary process learning

The main aim of incremental learning is the efficient usage of available resources when working with a data stream. The most important resources for the stationary process learning case are training time and storage space. The entire learning process will be reviewed regarding these parameters.

First of all, initial training data should be collected to start incremental learning. The sooner the quantum generator's state starts moving to the target, the earlier wanted relative entropy will be reached. On the other hand, optimizing a poor target state can diverge from a solution. Ordinarily, it is enough to collect only a few data batches, but the initial dataset size should be chosen according to the underlying process.

One of the main incremental learning features is updating a training dataset during training itself. After each update operation, the target state changes and loss functions gradients should be recalculated. Because of potentially endless data streams, it makes sense to define the stop training condition as reaching the wanted relative entropy value and check it after each training epoch and update operation. As soon as the target relative entropy is reached, the training process stops. It automatically continues in the case if another update operation increases relative entropy above the threshold.

With unlimited update potential, memory issue comes to the fore. Unlike the non-stationary process learning case, all provided data are relevant for the underlying process and, therefore, should be stored. For the problems with repeating data samples, e.g., probability distribution learning problem, a great solution is storing a frequency histogram instead of storing all provided samples. Such an approach significantly reduces using storage space and provides all the information as usual data stack.

Incremental learning modification of a quantum generative adversarial network provides an opportunity to specify the target state without losing previously passed data samples and trained generator's and discriminator's parameters. One of the main advantages over the original learning process is that there is no need to wait while collecting data from a data stream. It significantly reduces training time in real-world scenarios and provides impressive outcomes. Experiments results can be found in section 5.2.

3.5.2 Non-stationary process learning

Non-stationary process learning differs from the stationary one because underlying process parameters can change anytime, making collected samples irrelevant. Let data stream s sequentially provide data samples s_1, s_2, ..., s_n, where s_n is the latest available sample. The number of considered samples can be limited to predefined constant l, keeping in the memory only l latest samples: s_{n-l+1}, ..., s_n. The oldest data samples will be driven out by the new ones passed through the update operation. Such a technique resolves the irrelevant training data issue providing a reasonably chosen data stack size limit constant.

Unlike the stationary process learning case, time spent for initial training dataset collecting is not crucial for lifelong learning. The initial dataset size can be equal to the data stack limitation constant in the general case.

Lifelong learning of a non-stationary process aims to have a relevant quantum state at any learning time step. Therefore, the trained quantum state should be quickly adapted in a changing environment to keep relative entropy at an acceptable level. Thus an increased learning rate, as discussed in section 3.4, can be extremely helpful. Also, the stop condition based on the reached relative entropy can be used to prevent trained distribution divergence. The update interface of the implemented model allows to change the target relative entropy during training.

Chapter 4

Probability Distribution Learning Problem

This chapter describes the problem of loading random probability distribution into quantum state [7]. Underlying processes defined in sections 3.1.1 and 3.1.2 can be represented by some unknown probability distribution. Then the data stream contains samples generated from the unknown probability distribution. The sample generation process can be equivalently represented by measuring a quantum state because a quantum state has defined measuring probabilities for each basis state. Thus the problem of loading random probability distribution into a quantum state is quantum state approximation based on provided data samples from a data stream. Quantum state amplitudes and basis states should be adjusted to the underlying process, i.e., unknown probability distribution.

The loading random probability distribution problem is used to present the advantages of incremental learning for quantum generative adversarial network. However, described incremental learning features can be applied to any quantum state learning problem.

4.1 Definition

Let $X = \{x_1, ..., x_k\}$ be a list of each possible value of some discrete probability distribution, where event x_j occurs with probability $P(x_j)$. Of course, the normalization condition from equation 4.1 should be met.

$$\sum_{x \in X} P(x) = 1 \tag{4.1}$$

To represent generic discrete probability distribution with a quantum state, the number of its basis states 2^n, where n is the number of qubits, should be greater or equal to k. If the number of basis states is greater than k, then occurrence probabilities of unused basis states should be nullified. Then the problem can be defined as an approximation of a quantum state with corresponding probabilities.

$$|\Psi\rangle = \sum_{j=0}^{2^n-1} \sqrt{P(x_j)}\, |x_j\rangle \tag{4.2}$$

Equation 4.2 defines the target state, where $P(x_j)$ is a probability of occurrence of a basis state $|x_j\rangle$ representing an event $x_j \in X$. The aim of the quantum generative adversarial network training is to approximate this target state.

The number of basis states is defined by the number of qubits specified on the quantum generative adversarial network initialization. The number of qubits defines a quantum generator and cannot be changed during the learning process. Thus the dimension of a problem should be defined before the model's initialization and remain constant throughout the entire learning process.

The resulting state of a quantum variational circuit, i.e., quantum generator, can be modified by an optimizer changing the parameters of Pauli rotation gates. Then the target state approximation problem is reduced to optimizing a set of parameters θ.

$$G_\theta\, |\Psi_{in}\rangle = |\Psi\rangle \tag{4.3}$$

Equation 4.3 represents optimization target, where G_θ is the parametrized matrix of all quantum generator's gates and $|\Psi_{in}\rangle$ is initial quantum state. The initial quantum state is defined as $|0^n\rangle$ on most quantum devices, but theoretically can be any valid quantum state. The matrix G_θ can differ depending on initial probability distribution and the variational circuit architecture, but it should be parametrized.

4.2 Data

Described problem considers only discrete probability distributions. Since the underlying process is unknown, the probabilities of each state occurrence should be estimated from a training dataset. The more examples taken from a relevant underlying process are provided to the model, the more precisely the probabilities are calculated. In other words, relative entropy between the model's target and unknown distribution reduces as more samples become available.

$$D_{KL}(P \parallel Q) = \sum_{x \in \chi} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{4.4}$$

Relative entropy, also called Kullback-Leibler divergence [27], measures the informational distance between two probability distributions defined on the same probability space. Equation 4.4 defines relative entropy between probability distributions P and Q defined on the probability space χ.

Relative entropy is used for presenting testing results in chapter 5. There are three considered types of relative entropy in this thesis. “Relative entropy” refers to relative entropy between a probability distribution calculated from data samples generated by the actual quantum generator and a probability distribution calculated from the model’s training dataset. In other words, relative entropy shows the distance between actual and target quantum states. However, in case of incremental learning, the model’s target state is not constant and can be changed. “Real relative entropy” measures relative entropy between a probability distribution calculated from data samples generated by the actual quantum generator and the unknown target probability distribution that can be optionally defined for the experiment. “Data relative entropy” is relative entropy between a probability distribution calculated from the model’s training dataset and the unknown target probability distribution. The unknown target probability distribution is used only for receiving more information from experiments and is not used in the training process.

4.3 Applications

Probability distribution loaded into a quantum state can be used in the quantum amplitude estimation algorithm that provides a quadratic speedup compared to classical Monte Carlo methods. Any optimization problem based on the quantum amplitude estimation algorithm can use loaded probability distribution. The most interesting practical application based on the algorithm is option pricing [8].

Chapter 5

Results

Main contribution of this thesis is the implementation of a quantum generative adversarial network supporting incremental learning methods. The project was forked from the quantum generative adversarial network for learning and loading random distributions [7], and test cases are based on the probability distribution learning problem. However, the learning methods described in this thesis are applicable for any problem of a quantum state approximation using quantum generative adversarial networks.

5.1 Implementation

The project is divided into two parts: the implementation itself and the test environment. Both are available at the project's repository [23]. The implementation instruments are Python and its packages for machine learning (NumPy [21]) and quantum computing (Qiskit [22]).

5.1.1 Model

The implemented modification of a quantum generative adversarial network is compatible with all incremental and online learning scenarios. The model class is called `IQGAN` and exhibits the interface for incremental learning. This section provides a review of the most significant changes made to the original algorithm.

The stop training condition by the number of epochs was replaced with a required parameter of a target relative entropy. The parameter is called `target_rel_ent` and should be provided on the initialization. The training stops in case of relative entropy between trained quantum state and target dataset dropping below the provided target value. Because the target dataset is not constant, the learning process will automatically continue if relative entropy raises above the target value after the update operation. Furthermore, target relative entropy can be changed during learning according to the changing environment.

The original algorithm could calculate data bounds and data dimension based on provided complete training dataset. Because incremental learning does not suppose data availability at the beginning, data shape should be defined on the initialization manually. Moreover, updating a training dataset should follow the defined data shape because the model's architecture cannot be changed during the learning process. Thus the model's parameters `num_qubits` and `bounds` should be specified on the model's initialization.

The data stack size limit parameter called `max_data_length` can be optionally specified on the initialization. The parameter can be used for limiting memory complexity or learning concept drift underlying processes as described in section 3.5.2. Before reaching the limit size, the stack behaves ordinarily; after reaching the limit size, the oldest samples are replaced first.

The new option of storing training data as a frequency histogram instead of samples stack is introduced. This approach reduces memory complexity for many problems because every data sample is stored just once, and only its counter increases. This option is enabled by `freq_storage` parameter, but data samples should be comparable for storing with frequency histogram. The update operation also supports both storage policies.

The incremental model is much more dynamic than the original one, so it was updated to receive operations during the learning process. Multithreading allows the training process to be stopped or started anytime with new exposed operations: `stop_training` and `train`. The training stops at the end of the epoch after receiving the stop operation because of logging consistency and thread-safety reasons. The start operation resumes the learning process starting from the next epoch.

The new update operation expands the training dataset with the new data samples. The operation is called `update` and takes a dataset with new samples as an input. Optionally, the new target relative entropy can be specified. The data should be truncated and discretized to the bounds and the dimension before merging, this is done inside the function. Also, the update operation manages frequency histogram and limited data stack, if specified. Updating the training dataset affects the learning process only from the next epoch since the operation was received. The data will be shuffled before the learning continues. Otherwise, the training process mostly remains the same because of the model's external dependency on the quantum generator and classical discriminator. More about generator and discriminator setting up can be found in sections 3.2, 3.3, and 3.4.

In addition to important changes regarding the learning process, features for testing simplification were added as well. For example, unknown target dataset can be optionally provided on the initialization for relative entropy calculation between trained quantum state and unknown target. The parameter is called prob_data_real .The unknown target dataset is not used in the training process but only provides additional information about the experiment. Observation of real relative entropy makes sense in the context of incremental learning where target data are not precisely defined on the initialization, and thus standard relative entropy between trained quantum state and known training dataset does not reflect the real distance between actual and wanted quantum states. Also, the model logs its behavior and exports the log as a text file. Logging verbose parameter verbose is set up on the initialization.

The incremental model's interface is designed to be flexible and independent. Every input parameter and exposed operation are combining to cover all learning scenarios.

5.1.2 Test environment

The test environment for simplified experimentation was developed in addition to the model. The test environment initializes the model and conducts the learning process. The test environment defines operations for testing incremental quantum generative adversarial network on a probability distribution learning problem. Operations can be received in real-time through Python debugger input or via predefined test. Automatized tests are defined using the model's callback set up by set_training_handler function; after each epoch, the model passes the finished epoch number and actual relative entropy as parameters to the callback. Then, the callback can send operations to the model based on the epoch number and reached relative entropy. The complete list of operations can be found in the project's documentation.

Besides defined operations, the test environment logs the model's behavior and exports the log as a text file. Logging verbose parameter is set up on the model's initialization.

5.2 Tests

The most typical test cases for incremental learning of underlying processes from sections 3.5.1 and 3.5.2 are provided below. All experiments can be repeated with the test environment. The tests were performed with error-free quantum simulator provided by Qiskit. Tests with noisy quantum circuit simulator are given in appendix B.

Before running into obtained results, the data stream term should be formalized. Data steam $\mathcal{D} = (x_1, ..., x_n)$ is an ordered "first in, first out" queue with underlying process samples where sample x_{k+1} is obtained from the real data source quite after sample x_k. After taking top l samples from a data stream, they will be removed making $\mathcal{D} = (x_{l+1}, ..., x_n)$. Presented synthetic experiments use on-demand sample generation from probability distributions instead of raw data streams. Data stream with predefined data is represented by DataSource class in the project's code.

5.2.1 Stationary process learning

Underlying stationary process in the test case is represented by a log-normal probability distribution with mean $\mu = 1$ and sigma $\sigma = 1$. Figure 5.1 shows the distribution discretized and truncated into two bits version.

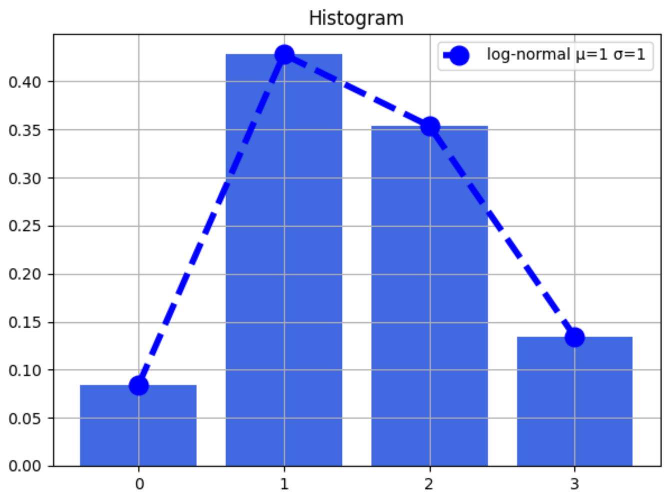

Figure 5.1: Histogram of discretized samples taken from log-normal distribution, $\mu = 1$ and $\sigma = 1$.

The test case is defined as following: the quantum generator is implemented with **RY** and **CZ** gates; the quantum generator's depth is 2; the quantum generator is initialized with fair uniform distribution; the discriminator's layers are defined with $2-50-20-1$ nodes; data stack size is unlimited; every data batch contains 5 samples; the model receives one batch per 10 time steps, i.e., learning epochs; learning rate equals to $1 \cdot 10^{-3}$; training starts with 1 data batch available.

Qubits number and batch size are limited due to faster computation but intended to represent real-world scenarios on a smaller scale. Figure 5.2 shows how relative entropy between the model's training data and unknown target reduces with new data passed to the model. It is seen that new training data passed at time steps 50 and 80 slightly increase relative entropy due to generation randomness, but for stationary underlying processes, data relative entropy tend to converge to zero when more samples become available.

New data passed to the model also impact relative entropy between the training dataset, i.e., target state, and samples generated by the variational circuit. This fact is reflected in figure 5.3 presenting relative entropy during learning. Obviously, the relative entropy changes according to the target state changing.

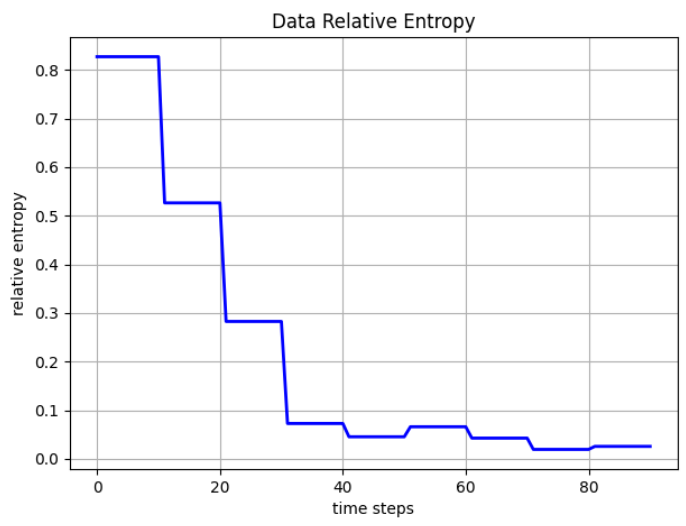

Figure 5.2: Data relative entropy for the stationary process learning test case.

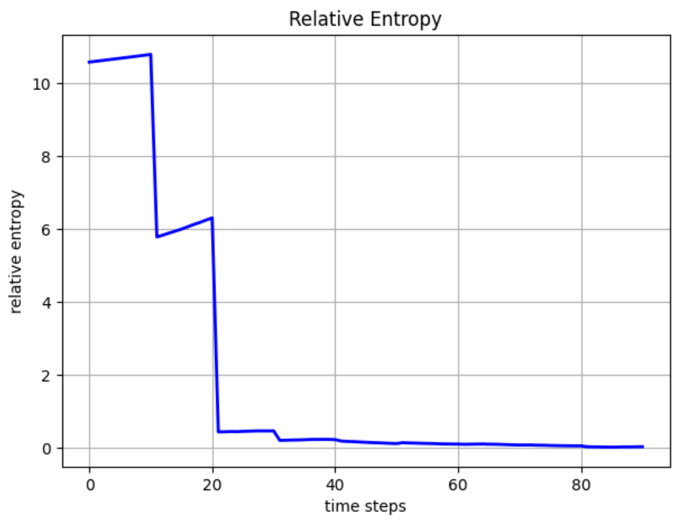

Figure 5.3: Relative entropy for the stationary process learning test case.

Relative entropy between unknown target and samples generated by the variational circuit, called real relative entropy, is much more representative because it shows convergence to the target across the whole learning process. Figure 5.4 shows the real relative entropy graph for the test case. Note that before time step 40, the model moves away from the target because of higher

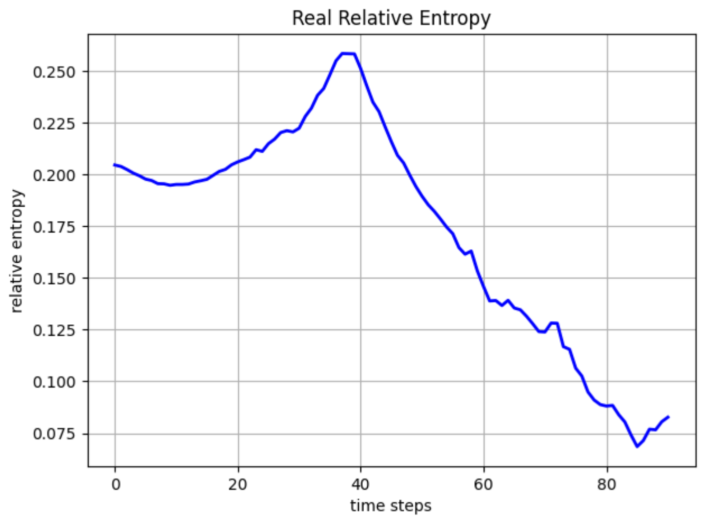

Figure 5.4: Real relative entropy for the stationary process learning test case.

data relative entropy. This behavior can be prevented by increased initial training data size, e.g., to 4 batches.

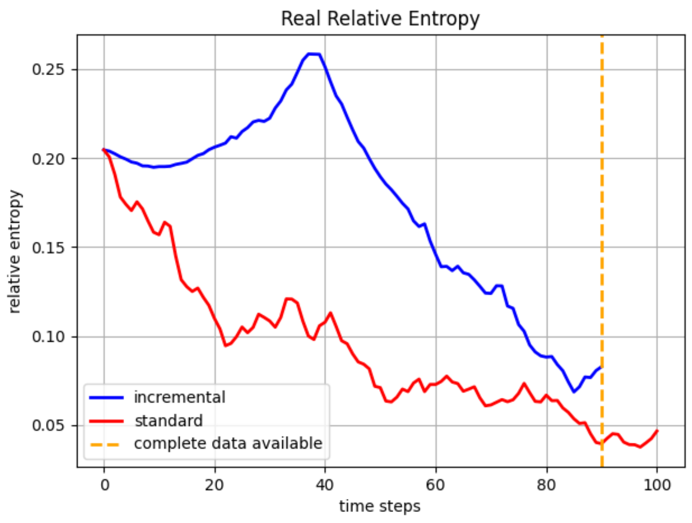

Figure 5.5: Comparison of incremental and original learning methods for the stationary process learning test case.

Comparing real relative entropy obtained from the incremental model with real relative entropy obtained from the original model helps evaluate the algorithm's efficiency. Figure 5.5 presents comparison results. The original quantum generative adversarial network was initialized the same way as the incremental one; the only difference is that the original model has all 10 data batches from the beginning, and the incremental one has initially only 1 data batch and receives another one per 10 passed time steps. Data samples were predefined and are identical between models to exclude randomness. The training process for the incremental model is finished at time step 90 because all data samples used for the original algorithm become available by the incremental test case definition. At this point, i.e., time step 90, data are collected and the original algorithm could be started. The advantage of the incremental learning model is that instead of being idle while collecting data, quantum generator parameters were trained, reducing real relative entropy from initial 0.2039 to much lower 0.0827.

5.2.2 Non-stationary process learning

Underlying non-stationary process for the test case is defined as a log-normal probability distribution with mean $\mu = 1$ and sigma $\sigma = 1$. Figure 5.1 from the previous section shows the distribution discretized into two bits. Because the process is non-stationary, let it be changed to a log-normal distribution with mean $\mu = 2$ and sigma $\sigma = 1$ at time step 100. Updated distribution is showed in figure 5.6.

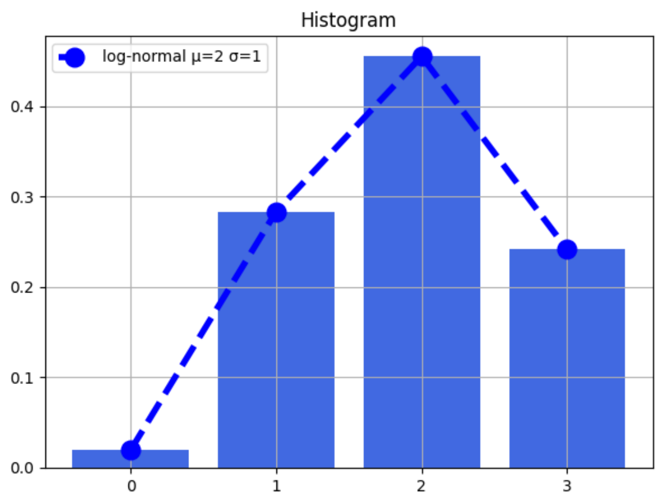

Figure 5.6: Histogram of discretized samples taken from log-normal distribution, $\mu = 2$ and $\sigma = 1$.

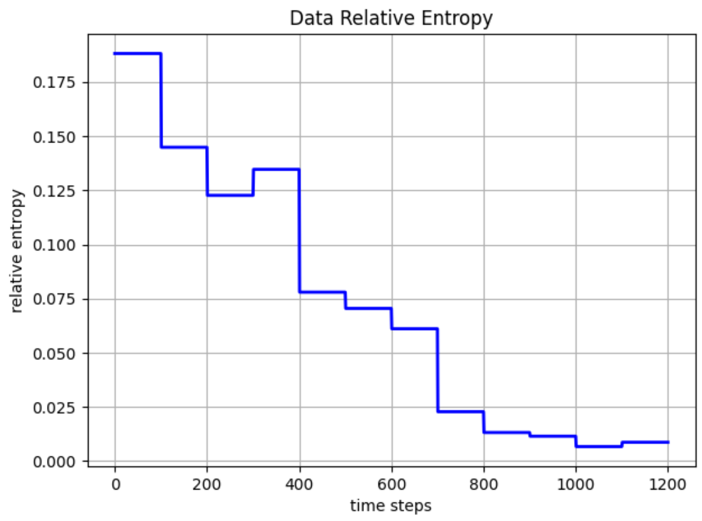

Figure 5.7: Data relative entropy for the non-stationary process learning test case.

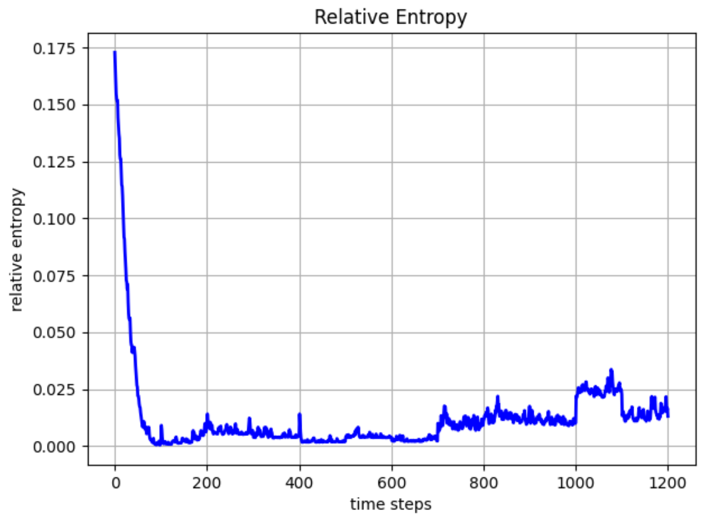

Figure 5.8: Relative entropy for the non-stationary process learning test case.

The test case is defined as following: the quantum generator is implemented with **RY** and **CZ** gates; the quantum generator's depth is 2; the quantum generator is initialized with fair uniform distribution; the discriminator's layers are defined with $2-50-20-1$ nodes; data stack size is limited to 100 samples; every data batch contains 10 samples; the model receives one batch per 100 time steps; learning rate equals to 10^{-3}; training starts with 10 data batches available.

Data stack size is limited because of the concept drift of the underlying distribution. Note that every data batch taken from the updated distribution after time step 100 and passed to the model will replace an irrelevant data batch in the model's training dataset. Figure 5.7 shows relative entropy changes between the model's training dataset and the unknown target represented by a log-normal distribution with mean $\mu = 2$ and sigma $\sigma = 1$.

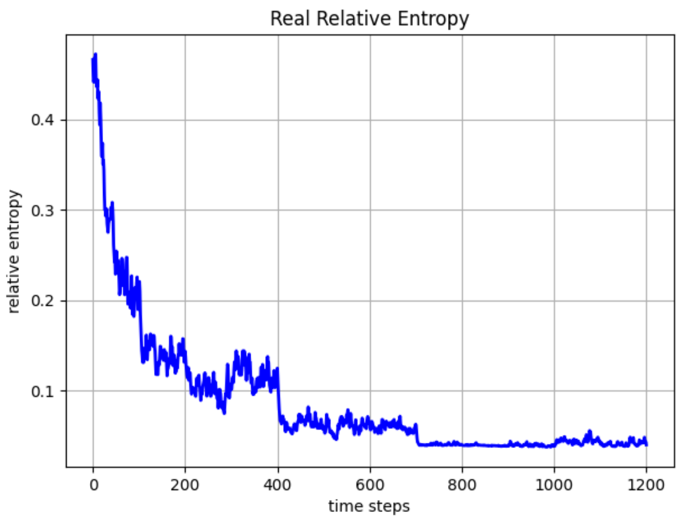

Figure 5.9: Real relative entropy for the non-stationary process learning test case.

By the test case definition, the model learns a log-normal distribution with mean $\mu = 1$ and sigma $\sigma = 1$ for the first 100 time steps and then receives the first data batch generated from the updated log-normal distribution with mean $\mu = 2$ and sigma $\sigma = 1$. At this point, the model's training dataset consists of 10% relevant and 90% irrelevant data. At time step 1000, the model's dataset consists of 100% relevant data.

Relative entropy between the trained and the target state is showed in figure 5.8. Note that the model is able to keep relevant quantum state during the whole learning process. Figure 5.9 presents real relative entropy between unknown target, i.e., log-normal distribution with mean $\mu = 2$ and sigma $\sigma = 1$, and the trained state.

The number of considered samples should be chosen according to the problem because it directly influences the time needed for transformation of a trained state, but it is clear that even with gradual updating the target state is reachable.

5.3 Applications

The algorithm still requires $\mathcal{O}(poly(n))$ gates for the quantum state preparation, but the training time for learning an underlying stationary process can be significantly reduced as follows from section 5.2.1. Training time is one of the most significant issues for using quantum generative adversarial networks for quantum state preparation, and incremental learning partially solves this problem.

On the other side, introduced lifelong learning of a non-stationary process can be used, e.g., in the quantum finance field. Prepared quantum state with probability distribution based on market data is used as an input for quantum amplitude estimation algorithm to calculate option pricing [8]. The lifelong learning of a non-stationary process can completely reduce the training time keeping always a relevant quantum state.

Moreover, designed incremental modification of quantum generative adversarial network can be applied on other problems than the probability distribution learning problem; thus, it can be used for any quantum state approximation based on some generic underlying process. Specific application examples are difficult to predict as the field of quantum computing is evolving rapidly.

Conclusion

Incremental learning scenarios for quantum generative adversarial networks for generic quantum state preparation were designed and described. The algorithm for incremental learning of quantum generative adversarial network was implemented and tested on the probability distribution learning problem. Also, the test environment for easier testing and experiment repetitions was developed.

The designed algorithm requires $\mathcal{O}(poly(n))$ gates for the quantum state preparation, the same as the original quantum generative adversarial network. However, the developed incremental approach significantly decreases training time for learning scenarios in which data from underlying processes become available gradually. Moreover, the implemented model can be used for lifelong learning of a non-stationary underlying process.

Possibly improvement of the implemented model could be training data management. Machine learning relies heavily on training data quality, especially it is important for incremental learning strategies. The target state can be defined poorly in case of lack of data representing the underlying process. Statistical analysis of training data, e.g., outliers detection, can probably solve this problem.

Another interesting continuation of this work could be applying incremental quantum generative adversarial networks to other problems than the probability distribution learning problem.

The implementation itself can be improved by making the frequency histogram storage policy compatible with the data stack size limitation. The improvement can be made by capturing the frequency histogram before its change via update operation. Combination of frequency histogram storage policy with limitation of considered samples number will significantly reduce memory complexity for learning concept drift processes.

The test environment also can be improved. Provided tests operate with the simplified term time step, i.e., time spent on a learning epoch. However, time spent on a learning epoch can differ depending on number of samples available for training. Thus the incremental model introduced in the stationary test case from section 5.2.1 actually receives the result earlier than its counterpart. Valuable improvement of the test environment can be supporting callbacks based on the real-time line instead of learning epochs.

Besides all mentioned improvements, there is a great potential for designing optimalization techniques for predefined parameters according to the problem. E.g., number of initially available data samples, batch size, learning rate, generator's and discriminator's architectures, and others.

Bibliography

[1] Arute, F.; Arya, K.; et al. Quantum supremacy using a programmable superconducting processor. *Nature*, volume 574, no. 7779, 2019: pp. 505–510.

[2] Plesch, M.; Brukner, Č. Quantum-state preparation with universal gate decompositions. *Physical Review A*, volume 83, no. 3, 2011: p. 032302.

[3] Brassard, G.; Hoyer, P.; et al. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, volume 305, 2002: pp. 53–74.

[4] Harrow, A. W.; Hassidim, A.; et al. Quantum algorithm for linear systems of equations. *Physical review letters*, volume 103, no. 15, 2009: p. 150502.

[5] Lloyd, S.; Weedbrook, C. Quantum generative adversarial learning. *Physical review letters*, volume 121, no. 4, 2018: p. 040502.

[6] Dallaire-Demers, P.-L.; Killoran, N. Quantum generative adversarial networks. *Physical Review A*, volume 98, no. 1, 2018: p. 012324.

[7] Zoufal, C.; Lucchi, A.; et al. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, volume 5, no. 1, 2019: pp. 1–9.

[8] Stamatopoulos, N.; Egger, D. J.; et al. Option pricing using quantum computers. *Quantum*, volume 4, 2020: p. 291.

[9] Yanofsky, N. S.; Mannucci, M. A. *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[10] Nielsen, M. A.; Chuang, I. Quantum computation and quantum information. 2002.

[11] Chamberland, C.; Noh, K.; et al. Building a fault-tolerant quantum computer using concatenated cat codes. *arXiv preprint arXiv:2012.04108*, 2020.

[12] Sych, D.; Leuchs, G. A complete basis of generalized Bell states. *New Journal of Physics*, volume 11, no. 1, 2009: p. 013006.

[13] Low, G. H.; Yoder, T. J.; et al. Quantum inference on Bayesian networks. *Physical Review A*, volume 89, no. 6, 2014: p. 062315.

[14] Bharti, K.; Haug, T.; et al. Machine learning meets quantum foundations: A brief survey. *AVS Quantum Science*, volume 2, no. 3, 2020: p. 034101.

[15] Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Goodfellow, I. J.; Pouget-Abadie, J.; et al. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[17] Michelsanti, D.; Tan, Z.-H. Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification. *arXiv preprint arXiv:1709.01703*, 2017.

[18] Huang, X.; Li, Y.; et al. Stacked generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5077–5086.

[19] Zhu, J.-Y.; Park, T.; et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[20] Paris, M.; Rehacek, J. *Quantum state estimation*, volume 649. Springer Science & Business Media, 2004.

[21] Harris, C. R.; Millman, K. J.; et al. Array programming with NumPy. *Nature*, volume 585, no. 7825, Sept. 2020: pp. 357–362, doi:10.1038/s41586-020-2649-2. Available from: https://doi.org/10.1038/s41586-020-2649-2

[22] Abraham, H.; AduOffei; et al. Qiskit: An Open-source Framework for Quantum Computing. 2019, doi:10.5281/zenodo.2562110.

[23] Kandaurov, A. Incremental learning of Quantum Generative Adversarial Network. https://github.com/kandaart/iqgan, 2021.

[24] Gepperth, A.; Hammer, B. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*, 2016.

[25] Reddi, S. J.; Kale, S.; et al. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[26] Murphy, K. P. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[27] Kullback, S.; Leibler, R. A. On information and sufficiency. *The annals of mathematical statistics*, volume 22, no. 1, 1951: pp. 79–86.

Appendix **A**

Influence of the learning rate parameter

Learning rate defines the step size at each iteration while moving toward a minimum of a loss function [26]. In practice, the learning rate is one of the most important parameters of the AMSGRAD optimizer used for the quantum generator and the classical discriminator in the test cases described in section 5.2. Different values of the parameter applied to both generator's and discriminator's optimizers were tested on the same test cases with the same data to exclude randomness.

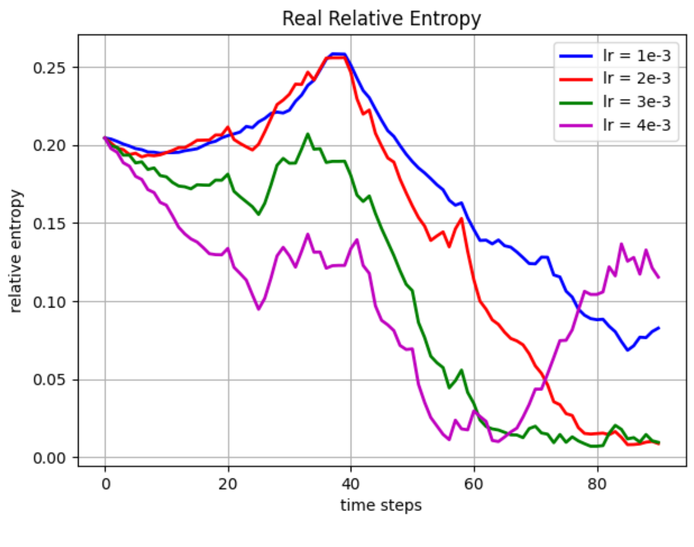

Figure A.1: Real relative entropy for the stationary process learning test case with variable learning rate parameter.

Figure A.1 shows the influence of the learning rate parameter on the real relative entropy for the test case defined in section 5.2.1. Notably, value $3 \cdot 10^{-3}$ provides the best convergence result for the given test case.

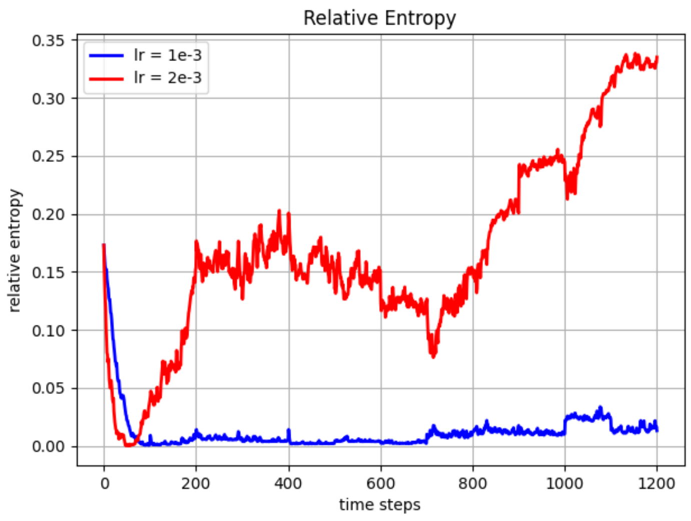

Figure A.2: Relative entropy for the non-stationary process learning test case with variable learning rate parameter.

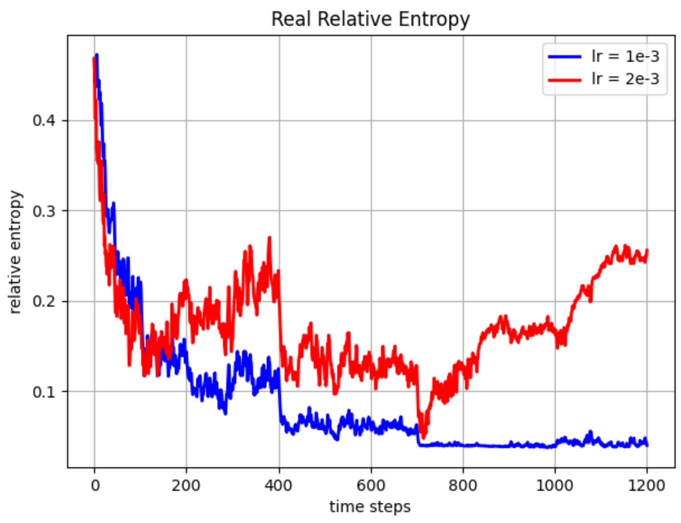

Figure A.3: Real relative entropy for the non-stationary process learning test case with variable learning rate parameter.

Figures A.2 and A.3 show the influence of the learning rate parameter for the test case defined in section 5.2.2. Increased learning rate provides significantly worse results for the test case as the trained state tends to diverge from the target.

Like in classical machine learning, there is no optimal parameters suitable for all problems. Learning rate should be chosen according to the problem because it highly depends on the underlying process complexity and the data stream definition.

Appendix B

Experiments with noisy quantum simulator

Test cases from section 5.2 were performed with a state vector simulator provided by Qiskit. The simulator is error-free, which is suitable for research but impossible for the actual quantum devices. Stationary process learning test case presented here was designed for noisy quantum simulator that is similar to a real quantum computer.

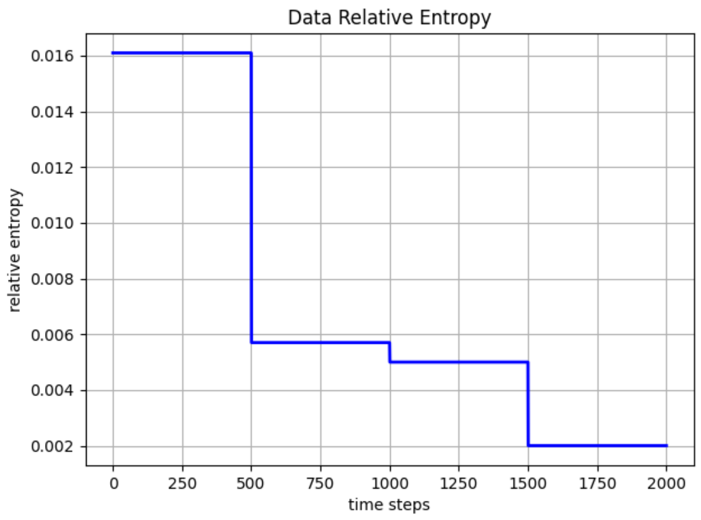

Figure B.1: Data relative entropy for the stationary process learning test case performed with a noisy quantum simulator.

The test case is defined as following: the quantum generator is implemented with **RY** and **CZ** gates; the quantum generator's depth is 2; the quantum generator is initialized with fair uniform distribution; the discrimi-

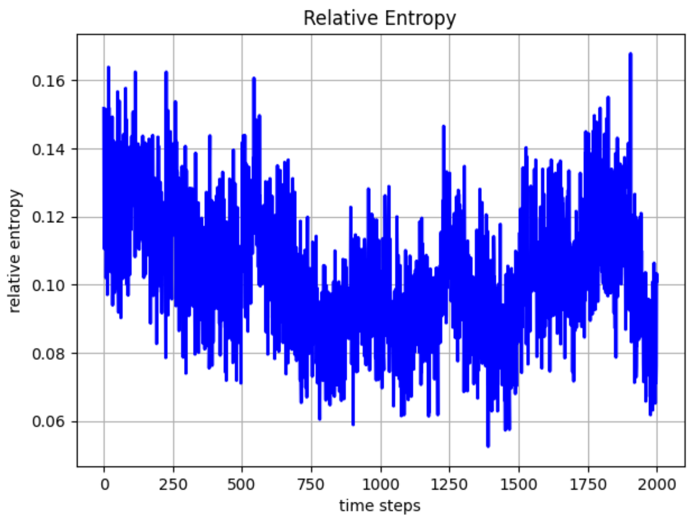

Figure B.2: Relative entropy for the stationary process learning test case performed with a noisy quantum simulator.

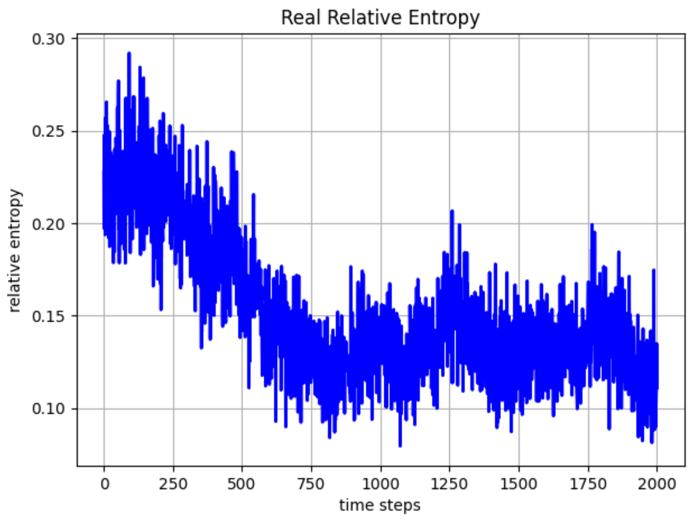

Figure B.3: Real relative entropy for the stationary process learning test case performed with a noisy quantum simulator.

nator's layers are defined with $2-50-20-1$ nodes; data stack size is unlimited; every data batch contains 500 samples; the model receives one batch per 500 time steps; learning rate equals to $1 \cdot 10^{-3}$; training starts with 1 data batch available. Underlying stationary process is represented by log-normal probability distribution with mean $\mu = 1$ and sigma $\sigma = 1$.

Appendix C

Contents of enclosed flash disk

```
thesis.pdf ........................................ the thesis text
└── iqgan ....................................... implementation sources
    ├── README.md
    ├── main.py
    ├── Algorithm
    │   ├── data_source.py ................. the data source implementation
    │   └── iqgan.py ........................... the model implementation
    ├── Test
    │   └── iqgan_uc_test.py .......... the test environment implementation
    └── Experiments
        ├── Stationary ............. stationary process learning experiments
        └── Non-Stationary ...... non-stationary process learning experiments
```