A Quantum-inspired Machine Learning based on The Heisenberg Uncertainty

Phuong-Nam Nguyen. Research Fellow, G.A.I.A QTech, Hai Ba Trung, Hanoi, 11600, Vietnam

Abstract—A quantum-inspired machine learning framework based on the Heisenberg uncertainty principle is derived and introduced in this article. First, we will give the geometric interpretation of the Heisenberg uncertainty picture. Second, the derived uncertainty relation is used to construct a machine learning model, named as HeisenbergBases. The computational experiments show that the proposed model performance is comparable to the conventional FNN, GRN, RNN and Transformer using three case studies. Moreover, a neural-based cell automata created from the HeisenbergBases is introduced and demonstrated in two use cases.

Index Terms—Physics-informed Machine Learning, Neural Networks, Quantum Mechanics, Mathematical Modeling, Simulation

Contents

1 Introduction 2

2 Framework 2
- 2.1 Geometric Interpretation of The Heisenberg Uncertainty 2
- 2.2 Derivation of Parameterized Uncertainty 3
- 2.3 Our Postulation 5
- 2.4 Model: HeisenbergBases 5

3 Case Studies 6
- 3.1 Experiment Design 6
- 3.2 Quantum State Learning 6
- 3.3 Applications in Material Science 6
  - 3.3.1 Estimation of Temperature-Solubility Relation 6
  - 3.3.2 Spectral Classifications of Materials 6
- 3.4 Cell Automata 7
  - 3.4.1 Dynamical Simulation for Two Competitive Creatures . 8
  - 3.4.2 Simulation of Hydrogen's Electron Orbit 9

4 Discussion 9
- 4.1 The Link to the Heisenberg Uncertainty 9
- 4.2 Difference from Classical Neural Networks 9
- 4.3 Future Works 9
  - 4.3.1 Scalability of the model 9
  - 4.3.2 Further applications 10
  - 4.3.3 Limitations 10

5 Conclusion 10

References 10

6 Appendix 10

Abbreviations

Models

FNN — Feed-forward Neural Network(s)
GRN — Gated Relation Networks(s)
ML — Machine Learning
NN — Neural Networks
PIML — Physics-Informed Machine Learning
QIML — Quantum-Informed Machine Learning
QML — Quantum Machine Learning
RNN — Recurrent Neural Networks(s)

Mathematical Notations

O_{xyz} — 3D coordinate
u — unit length
$A, B, C, \ldots$ — point(s)
$OA, OB, OC, AB, \ldots$ — Segment(s)
$\mathcal{M}$ — Manifold
S — Area
$\phi, \psi, \kappa, \Psi, \alpha, \ldots$ — function(s)
$x, p, t, \ldots$ — Variable(s)
$\mathbb{E}$ — Expectation
μ — mean
σ — standard deviation
Λ — Array of bases
$\mathbf{h}$ — Latent embeddings
$\mathbf{X}, \mathbf{y}$— Input data
Re, Im — Real and imaginary part of comple number

1 Introduction

Quantum Computing is a cutting-edge field that leverages the principles of quantum mechanics to perform computations far beyond the capabilities of classical computers. Quantum computers utilize qubits, which can exist in multiple states simultaneously (superposition) and are interconnected through quantum entanglement. This allows for parallel processing at an unprecedented scale, offering the potential to revolutionize fields like cryptography, optimization, and material science.

Physics-Informed Machine Learning (PIML) integrates physical laws and principles into machine learning models to enhance predictive capabilities and improve generalization. By incorporating constraints from fields like quantum mechanics, fluid dynamics, or thermodynamics, PIML ensures that models adhere to known physical phenomena, making them more robust and interpretable in applications such as scientific computing, engineering, and environmental modeling.

The hotly debated literature of quantum machine intelligence could be summarized with exacerbation as three main directions:

1) Implement classical algorithms on quantum computers.
2) Using classical algorithms as complementary to help quantum algorithms. In my previous work, a circuit architecture search algorithm for the quantum entangling layout was developed, which searches using the graphical representations of such a layout, named QES [5].
3) Develop quantum algorithms for quantum computers: Exploiting the statistical property of the quantum wavefunction, I develop a Bayesian classifier (Bayesian-QNN) for classification tasks [4]. Besides, a game-theoric framework for dynamical modeling is introduced in [6], named the Duality Game. The model is then extended for biomarker identifications, a significant medical problem. The later focus is on encoding data efficiently on quantum computing, resulting in [7], [8].

The main theme of this article is Quantum-inspired machine learning (QIML), a paradigm where the physical laws of quantum mechanics constrain models. QIML differs from QML as the former is a classical neural network inspired by quantum mechanics, while the latter is a machine learning algorithm on quantum computers. QIML differs from classical NN since the model weights (degree of freedom) are constrained to represent the physics of quantum systems.

Our contribution is three-fold:

1) We will propose a new PIML framework that develops machine intelligence grounded in quantum physics;
2) We will develop AI models inspired by the Heisenberg uncertainty principle, referred to as HeisenbergBases;
3) The proposed model is powerful for representation learning, demonstrating superiority over classical neural network architectures, as numerically validated in three case studies.

The setting for numerical demonstration is designed for low-limited datasets, i.e., a small number of observations. The proposed model is generally compatible with or better than classical neural architecture such as FNN, GRN, RNN, and Transformer models.

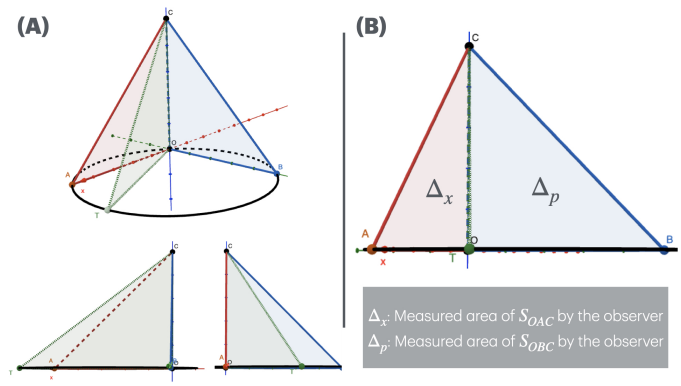

Fig. 1: Geometric Interpretation of The Heisenberg Uncertainty

This article is organized as follows:

- Section 2 introduces our proposed framework of QIML based on the Heisenberg uncertainty principle of quantum mechanics. We will start with the geometric interpretation of this uncertainty principle in Section 2.1. Then, we will give an implementation of AI models using our derivatives in Section 2.4.
- Section 3 report the performance of our model compared with FNN, GRN, RNN, and Transformer models in three case studies: (1) quantum state learning, (2) estimation of temperature-solubility in Al and Cu, (3) spectral classification of materials. We also introduce how to create a cell automaton from the Heisenberg-Bases model.
- Section 4 discusses the link of our derivation to the original Heisenberg picture. Then, we will emphasize how our proposed model differs from the classical NN. Finally, several future directions are discussed.

2 Framework

2.1 Geometric Interpretation of The Heisenberg Uncertainty

Given a pyramid in $Oxyz$ coordinate with edge length $OA = OB = OC = u$. Imagine an observer moves along the side AB looking straight into the origin O, and their position on AB is a point T (Fig. 1(A), top panel). The observation plane is OCT. In this setting, the observer can measure the area of the two triangles

$$\begin{aligned} S_{OAC} &= \frac{1}{2} OA * OC = 0.5u \times OB \\ S_{OBC} &= \frac{1}{2} OB * OC = 0.5u \times OA \end{aligned} \tag{1}$$

The perceived area by the observers changes due to the coordinate of T. For example in the bottom panel of Fig. 1(A):

- $T \equiv A$, the observer fully see the plane Oyz and so the area of Δ_p is fully measured. When the observer is at point A, they have a complete view of triangle OBC, meaning that S_{OBC} is fully measured while S_{OAC} is not.

- $T \equiv B$, the observer fully see the plane Oxz and so the area of Δ_x is fully measured. When the observer is at point B, they fully see triangle OAC, meaning S_{OAC} is fully measured, but S_{OBC} is not.
- When T moving from A to B, the area of Δ_x increases while that of Δ_p decreases.

The observer cannot simultaneously fully measure Δ_p and Δ_x. As the observer moves from A to B, the measured area of S_{OAC} increases while that of S_{OBC} decreases, much like the uncertainty relationship: a more precise measurement of one triangle corresponds to a less precise measurement of the other.

This mirrors the idea of uncertainty in quantum systems, where the precise measurement of one variable (e.g., position) leads to increased uncertainty in the measurement of its complementary variable (e.g., momentum). In the geometrical analogy, the observer's inability to fully measure both areas simultaneously represents the inherent trade-off between precision in complementary measurements.

2.2 Derivation of Parameterized Uncertainty

Let $\mathcal{M}$ be the manifold parameterized by (x, y, t) denoted as $M(x, y, t)$. We consider $M(x, p, t)$ to link with the Heisenberg uncertainty. We denote

- $\phi_{AC} = \phi(x, t)$ as a function obtained by project M onto Oxz.
- $\psi_{BC} = \psi(p, t)$ as a function obtained by project M onto Oyz.

The area under the curve $\phi(x, t)$ and $\psi(p, t)$ are the area of Δ_x and Δ_p; respectively, measured by the observers (Fig. 1(B)).

We have

$$\begin{aligned} \Delta_x &= \int_{\Omega_x} \phi(x,t)dt \\ \Delta_p &= \int_{\Omega_y} \psi(p,t)dt \end{aligned} \tag{2}$$

When $OA = OB$, the upper bound of the integrals is the same, which implies $\Omega_x = \Omega_y = \Omega$. We can rewrite

$$\Delta_x + \Delta_p = \int_{\Omega} \phi(x,t)dt + \int_{\Omega} \psi(p,t)dt \tag{3}$$

Let

$$\begin{aligned} \phi(x,t) &= x^2(t)\kappa(x|p,t) \\ \psi(p,t) &= p^2(t)\kappa(p|x,t) \end{aligned} \tag{4}$$

where $\kappa(x, p, t)$ is the multivariate distribution of (x, p, t). The conditional probability of x (position) and p (momentum) being measured are $\kappa(x|p, t)$ and $\kappa(p|x, t)$, respectively. Plugging-in Equa. 3, we have

$$\begin{aligned} \eta &:= \Delta_x + \Delta_p \\ \eta &= \int_{\Omega} x^2(t)\kappa(x|p,t)dt + \int_{\Omega} p^2(t)\kappa(p|x,t)dt \end{aligned} \tag{5}$$

Here, we assume that $x(t)$ and $p(t)$ are random variables changes w.r.t time t. Besides, we set

$$\begin{aligned} (A_1) &: \kappa(x|t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2(t)}{2}\right) \\ (A_2) &: \kappa(p|t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{p^2(t)}{2}\right) \\ (A_3) &: \kappa(t) = \frac{1}{u} \quad \text{for } t \in [0, u] \end{aligned} \tag{6}$$

(A_1) and (A_2) assume that x and p are random variables from $\Phi(0, 1)$ and (A_3) assumes t is from $U[0, u]$. If $x(t)$ and $p(t)$ are independent, their joint distribution is

$$\kappa(x, p|t) = \frac{1}{2\pi} \exp\left(-\frac{x^2(t)}{2}\right) \exp\left(-\frac{p^2(t)}{2}\right) \tag{7}$$

Moreover, x and p are depend on t by relations $x(t)$ and $p(t)$. Using the Bayesian rule, we obtain

$$\kappa(x, p|t) = \frac{\kappa(x, p, t)}{\kappa(t)} = u\kappa(x, p, t) \tag{8}$$

or

$$\kappa(x, p, t) = \frac{1}{2u\pi} \exp\left(-\frac{x^2(t)}{2}\right) \exp\left(-\frac{p^2(t)}{2}\right). \tag{9}$$

Similarly, we have

$$\begin{aligned} \kappa(x,t) &= \kappa(x|t)\kappa(t) = \frac{1}{u\sqrt{2\pi}} \exp\left(-\frac{x^2(t)}{2}\right) \\ &= \frac{\gamma}{u} A(x,t) \\ \kappa(p,t) &= \kappa(p|t)\kappa(t) = \frac{1}{u\sqrt{2\pi}} \exp\left(-\frac{p^2(t)}{2}\right) \\ &= \frac{\gamma}{u} A(p,t) \end{aligned}, \tag{10}$$

where $\gamma = 1/\sqrt{2\pi}$ and $A(\circ_1, \circ_2) = \exp\left(-\frac{\circ_1(\circ_2)^2}{2}\right)$. The conditional probability in Equa. 5 is

$$\begin{aligned} \kappa(x|p,t) &= \kappa(x,p,t)\kappa(p,t) = \frac{\gamma^3}{u^2} A(x,t)A^2(p,t) \\ \kappa(p|x,t) &= \kappa(x,p,t)\kappa(x,t) = \frac{\gamma^3}{u^2} A^2(x,t)A(p,t) \end{aligned} \tag{11}$$

We have

$$\begin{aligned} \Delta_x &= \int_0^u x^2(t) \frac{\gamma^3}{u^2} A(x,t)A^2(p,t) \\ &= \frac{\gamma^3}{u^2} \int_0^u [A(x,t)A(p,t)]A(p,t)x^2(t) \\ \Delta_p &= \int_0^u p^2(t) \frac{\gamma^3}{u^2} A^2(x,t)A(p,t) \\ &= \frac{\gamma^3}{u^2} \int_0^u [A(x,t)A(p,t)]A(x,t)p^2(t) \\ A(x,t)A(p,t) &= 2u\pi\kappa(x,p,t) \\ \implies \Delta_x &= \frac{2\pi\gamma}{u} \int_0^u \kappa(x,p,t)A(p,t)x^2(t)dt \\ \implies \Delta_p &= \frac{2\pi\gamma}{u} \int_0^u \kappa(x,p,t)A(x,t)p^2(t)dt \end{aligned} \tag{12}$$

Simplifying the term Δ_x and Δ_y, we have

$$\begin{aligned}\Delta_x &= \frac{\sqrt{2\pi}}{u}\int_0^u \kappa(x,p,t)A(p,t)x^2(t)dt \\ \Delta_p &= \frac{\sqrt{2\pi}}{u}\int_0^u \kappa(x,p,t)A(x,t)p^2(t)dt\end{aligned} \tag{13}$$

Since u is just the unit of $Oxyz$ (recalled that $OA = OB = OC = u$), we can set $u = 1$ and the term

$$\begin{aligned}&\Delta_x\Delta_p = 2\pi\int_0^u \kappa(x,p,t)B_x dt \cdot \int_0^u \kappa(x,p,t)B_p dt \\ &B_x(t) = A(p,t)x^2(t) \\ &B_p(t) = A(x,t)p^2(t)\end{aligned} \tag{14}$$

Clearly, B_x and B_p are random variables, assumed that B_x and B_p from the same distribution $\kappa(x,p,t)$; then, the integral returns the expectation of E_x and E_p, says

$$\begin{aligned}\mathbb{E}(B_x) &= \int_0^u \kappa(x,p,t)B_x(t)dt \\ \mathbb{E}(B_p) &= \int_0^u \kappa(x,p,t)B_p(t)dt\end{aligned} \tag{15}$$

As a result, we have

$$\Delta_x\Delta_p = 2\pi\mathbb{E}(B_x)\cdot\mathbb{E}(B_p). \tag{16}$$

and

$$\begin{aligned}\mathbb{E}(B_x) &= \int_0^u \frac{1}{2u\pi}\exp\left(-\frac{x^2(t)+p^2(t)}{2}\right)A(p,t)x^2(t)\,dt \\ \mathbb{E}(B_p) &= \int_0^u \frac{1}{2u\pi}\exp\left(-\frac{x^2(t)+p^2(t)}{2}\right)A(x,t)p^2(t)\,dt\end{aligned}, \tag{17}$$

which directly implies

$$\begin{aligned}\mathbb{E}(B_x) &= \int_0^u \frac{1}{2u\pi}\exp\left(-\frac{x^2(t)+2p^2(t)}{2}\right)x^2(t)\,dt \\ \mathbb{E}(B_p) &= \int_0^u \frac{1}{2u\pi}\exp\left(-\frac{2x^2(t)+p^2(t)}{2}\right)p^2(t)\,dt\end{aligned}, \tag{18}$$

and

$$\begin{aligned}&\Delta_x\Delta_p = \frac{1}{u}\mathbb{E}(B_x)\cdot\mathbb{E}(B_p) \\ &= \int_0^u C_1(x(t),p(t))x^2(t)\,dt \cdot \int_0^u C_2(x(t),p(t))p^2(t)\,dt \\ &C_1(x(t),p(t)) := \exp\left(-\frac{x^2(t)+2p^2(t)}{2}\right), \\ &C_2(x(t),p(t)) := \exp\left(-\frac{2x^2(t)+p^2(t)}{2}\right)\end{aligned}. \tag{19}$$

Applying the Cauchy-Schwarz inequality, we have

$$\Delta_x\Delta_p \geq \int_0^u x(t)p(t)\sqrt{C_1C_2}dt, \tag{20}$$

where

$$\begin{aligned}C_1C_2 &= \exp\left(-\frac{x^2(t)+2p^2(t)+2x^2(t)+p^2(t)}{2}\right) \\ &= \exp\left(-\frac{3x^2(t)+3p^2(t)}{2}\right) \\ &= \exp\left(-\frac{3(x^2(t)+p^2(t))}{2}\right)\end{aligned} \tag{21}$$

and

$$\sqrt{C_1C_2} = \exp\left(-\frac{3(x^2(t)+p^2(t))}{4}\right). \tag{22}$$

The normalization of $\sqrt{C_1C_2}$ is computed by

- Compute the integral of $\exp\left(-\frac{3(x^2+p^2)}{4}\right)$ over all x and p:

  $$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\exp\left(-\frac{3(x^2+p^2)}{4}\right)dx\,dp$$

  This integral can be separated into two Gaussian integrals:

  $$\left(\int_{-\infty}^{\infty}\exp\left(-\frac{3x^2}{4}\right)dx\right)\times\left(\int_{-\infty}^{\infty}\exp\left(-\frac{3p^2}{4}\right)dp\right)$$

  Each integral is a Gaussian integral of the form:

  $$\int_{-\infty}^{\infty}\exp\left(-\alpha x^2\right)dx = \sqrt{\frac{\pi}{\alpha}}$$

- Here, $\alpha = \frac{3}{4}$. Therefore:

  $$\int_{-\infty}^{\infty}\exp\left(-\frac{3x^2}{4}\right)dx = \sqrt{\frac{4\pi}{3}}$$

  $$\int_{-\infty}^{\infty}\exp\left(-\frac{3p^2}{4}\right)dp = \sqrt{\frac{4\pi}{3}}$$

  So, the total integral is:

  $$\left(\sqrt{\frac{4\pi}{3}}\right)\times\left(\sqrt{\frac{4\pi}{3}}\right) = \frac{4\pi}{3}$$

- To normalize $\sqrt{C_1C_2}$, divide it by this integral:

  $$\text{Normalized } \sqrt{C_1C_2} = \frac{\exp\left(-\frac{3(x^2+p^2)}{4}\right)}{\frac{4\pi}{3}}$$

  Simplify the fraction:

  $$\text{Normalized } \sqrt{C_1C_2} = \frac{3}{4\pi}\exp\left(-\frac{3(x^2+p^2)}{4}\right)$$

We denote Normalized $\sqrt{C_1C_2} = \Psi(x,p,t)$, then

$$\exp\left(-\frac{3(x^2+p^2)}{4}\right) = \frac{4\pi}{3}\Psi(x,p,t) \tag{23}$$

and the Equa. 20 becomes

$$\begin{aligned}\Delta_x\Delta_p &\geq \int_0^u x(t)p(t)\sqrt{C_1C_2}dt \\ &= \frac{4\pi}{3}\int_0^u x(t)p(t)\Psi(x,p,t)dt.\end{aligned} \tag{24}$$

Of note, the multivariate probability Ψ is the density function due to the normalization, i.e., it is some joint distribution of (x,p,t). Thus, the integral can be given as

$$\Delta_x\Delta_p \geq \frac{4\pi}{3}\mathbb{E}_\Psi[x(t)p(t)] \tag{25}$$

Geometric Relation Between $x(t)$ and $p(t)$: The quantities $x(t)$ and $p(t)$ can be projected onto Oxy, say the

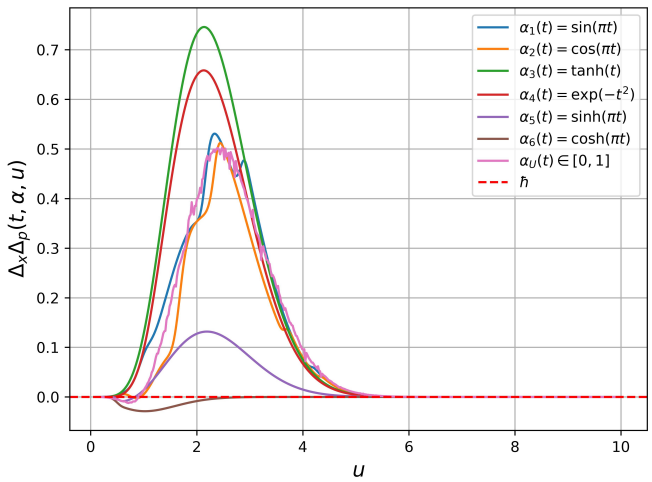

Fig. 2: The value of $\Delta_x\Delta_p(t, \alpha, u)$ w.r.t unit of measurement $u \in [0, 1]$, $\alpha(t)$ is a bounded function in $[0, 1]$ and $t \in [0, u]$

projected point of $H(x, 0, u)$ and $Q(0, p, u)$ are $P(x, 0, 0)$ and $N(p, 0, 0)$. Besides,

- $|OP| = (1 - \alpha(t))u$ and
- $|ON| = \alpha(t)u$, where $\alpha(t) \in [0, 1]$.

Therefore,

$$x(t)p(t) = u^2\alpha(t)(1 - \alpha(t)) \tag{26}$$

$$\begin{aligned} \Delta_x\Delta_p &\geq \int_0^u u^2\alpha(t)(1 - \alpha(t))\sqrt{C_1C_2}dt \\ &= u^2 \int_0^u \alpha(t)(1 - \alpha(t)) \exp\Big(-\frac{3}{4}r^2\Big)dt \\ r &= x^2(t) + p^2(t) \end{aligned} \tag{27}$$

Given the expressions for $x(t)$ and $p(t)$ as:

$$x(t) = (1 - \alpha(t))u, \quad p(t) = \alpha(t)u,$$

We denoted the **modified Heisenberg relation** $\Delta_x\Delta_p(t, \alpha, u)$, given as

$$\begin{aligned} \Delta_x\Delta_p(t, \alpha, u) &= u^2 \int_0^u \alpha(t)(1 - \alpha(t))D(t, \alpha)dt, \\ D(t, \alpha) &= \exp\left(-\frac{3u^2}{4}\Big[\alpha(t)^2 + (1 - \alpha(t))^2\Big]\right) \end{aligned} \tag{28}$$

where:

- $\alpha(t)$ is a bounded function in $[0, 1]$,
- $t \in [0, u]$

2.3 Our Postulation

In the Heisenberg uncertainty, the product $\Delta_x\Delta_p \geq \hbar/2$ is a constant lower bound. Here, we postulate that this product is a function of time t, the time-dependent action $\alpha(t)$ bounded in $[0, 1]$ and unit of measurements u. We test seven bounded function for α, given in Fig. 2, including $\sin, \cos, \tanh, \exp(-t^2), \sinh, \cosh$ and $\alpha \in U[0, 1]$ (uniformly distributed). There are three main observations:

1) Not all bounded function α gives a physical sense. Specifically, using $\alpha_6(t) = \cosh(\pi t)$ result in negative $\Delta_x\Delta_p$. In fact, $\Delta_x > 0$ and $\Delta_p > 0$.

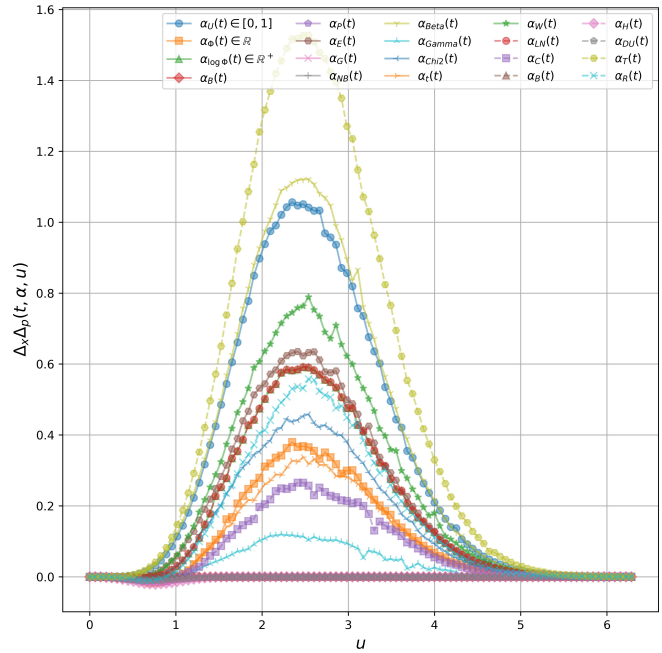

Fig. 3: The value of $\Delta_x\Delta_p(t, \alpha, u)$ from our proposed probabilistic sampler

2) There exist classes of functions with positive $\Delta_x\Delta_p$. Specifically, other α functions (except α_6) satisfies the above physical constraint, so as scaled α functions by a positive scalar.
3) When α in a random variable from the uniform distribution $[0, 1]$, we find that $\Delta_x\Delta_p$ not only has lower bound $\hbar/2$ as the Heisenberg uncertainty principle but also has the upper bound, showed as α_U in Fig. 2.

These findings contribute to understanding how bounded time-dependent actions influence the uncertainty product, showing that specific classes of functions for $\alpha(t)$ can maintain the physical constraints of the Heisenberg uncertainty principle while revealing the existence of an upper bound in certain cases.

2.4 Model: HeisenbergBases

This section will introduce a neural network model based on the class of functions $\Delta_x\Delta_p(t, \alpha, u)$. It is observed that the values of the Δ function depend on the generating probability distribution for $\alpha(t)$. We create the array

$$\Lambda_{N_b \times d} = [\Delta_x\Delta_p(\alpha_1), \Delta_x\Delta_p(\alpha_2), \ldots, \Delta_x\Delta_p(\alpha_{N_b})], \tag{29}$$

using the same values of t and u. The summary of all generating distribution functions α is reported in Tab. 6.

Given an input matrix $\mathbf{X}_{N \times D} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D]$ where N is the number of samples and D is the number of features, we compute the embedding

$$\mathbf{h}_{N \times D} = \frac{1}{D}\sum_i^D \frac{1}{N_b}\sum_j^{N_b} \Delta_x\Delta_p(\alpha_j(t)) \cdot \mathbf{x}_i. \tag{30}$$

The classification layer is a common MLP $f_\Theta(\mathbf{h})$, which gives the prediction $\hat{\mathbf{y}} = f_\Theta(\mathbf{h})$. We name this neural network HeisenbergBases since it is based on our derived uncertainty constraint.

3 Case Studies

3.1 Experiment Design

The input data is $\mathbf{X}$ for each following experiment. First, we split the data into two independent train and test sets (with ratio $80\%, 20\%$), with the statistics summarized in Tab. 1. The number of observations is small, similar to practical scenarios.

We train a model $f_\Theta(x)$ to reach the minumum of six objective functions $\mathcal{L}$: (1) MSE, (2) L1 Loss, (3) smooth L1, (4,5,6): Huber Loss of $\delta = 1, 2, 3$. We evaluate how well these models can re-construct the original signals. Specifically, we aim to solve

$$\Theta^* = \arg\min_\Theta \mathcal{L}(\hat{\mathbf{y}}, \mathbf{X}) \tag{31}$$

The smaller loss means the closer distance between the predicted vector $\mathbf{y}$ and the input $\mathbf{X}$, so a good model is obtained. We build the model using `torch`, and the hyper-parameter is performed by `optuna`. Each neural architecture is trained using 50 epochs under 10 trials of hyper-parameter optimization. We use AdamW [3] optimizer with a learning rate of 0.005. Random seeds are set for each run. All experiments were conducted with Python 3.7.0 using an Intel i9 processor (2.3 GHz, eight cores) and 16GB DDR4 RAM.

3.2 Quantum State Learning

The Wigner function for a coherent state is given by:

$$W(q,p;\lambda) = \frac{2}{\pi}\exp\left(-2\left((q-\mathrm{Re}(\alpha))^2 + (p-\mathrm{Im}(\lambda))^2\right)\right) \tag{32}$$

The Wigner function for the superposition of two coherent states (Schrödinger's cat state) is defined as:

$$\begin{aligned} W_{\text{cat}}(q,p;\lambda) = &\frac{1}{2}\left(W(q,p;\lambda) + W(q,p;-\lambda)\right) \\ &+ 2\exp(-2(q^2+p^2))\cos(4p\,\mathrm{Re}(\lambda)) \end{aligned} \tag{33}$$

We use $\lambda = 1$ for the synthesized data.

The performance of different models is compared using hypothesis tests with the quantum cat state learning task, reported in Fig 4.

For FNN vs. HeisenbergBases, the p-value is 0.1398, which is higher than 0.05, indicating no significant difference in performance between these models. In the comparison of GRN vs. HeisenbergBases, the p-value is 0.6504, again showing no significant difference. However, for RNN vs. HeisenbergBases, the p-value is 0.0206, which is below the 0.05 threshold, indicating that HeisenbergBases performs significantly better than the RNN model. Lastly, the Transformer vs. HeisenbergBases comparison yields a p-value of 0.3082, indicating no significant difference in their performance. HeisenbergBases shows relatively competitive performance with FNN and GRN across several loss functions. HeisenbergBases exhibits low variability (low standard deviation) compared to all other models ((Fig 4(B))). It has a higher variance than GRN across most loss functions, while RNN and Transformer show lower variance than HeisenbergBases. In AppTab. 2, we have enough statistical evidence to show that HeisenbergBases's performance is better than RNN and Transformer, comparable to FNN and GRN.

3.3 Applications in Material Science

3.3.1 Estimation of Temperature-Solubility Relation

BigSolDB [2] contains $54,273$ solubility values, 830 unique molecules, and 138 individual solvents within the temperature range from 243.15 to $403.15K$ at atmospheric pressure. The dataset consists of six columns, which are explained as follows:

- **SMILES** — SMILES representation of a dissolved compound
- **T,K** — Temperature in Kelvin
- **Solubility** — Experimental solubility value (mole fraction)
- **Solvent** — Name of the solvent
- **SMILES_Solvent** — SMILES of the solvent
- **Source** — A data source for the given values

We use the temperature and solubility values for the model evaluation.

In Tab. 3, the performance of different models is compared using hypothesis tests on the *BigSol* dataset.

For FNN vs. HeisenbergBases, the p-value is 0.9995, indicating no significant difference in performance between these models. Similarly, the comparison of GRN vs. HeisenbergBases gives a p-value of 0.9999, showing no significant difference. However, for the comparison of HeisenbergBases vs. RNN, the p-value is 1.79×10^{-5}, which is below the 0.05 threshold, indicating that the HeisenbergBases model significantly outperforms the RNN. For Transformer vs. HeisenbergBases, the p-value is 0.0004, also indicating a significant difference, with the HeisenbergBases model performing better. Overall, HesenbergBases is the best model in this case study, significantly outperforming all others.

3.3.2 Spectral Classifications of Materials

Spectral data from aluminum, stainless steel, mild steel, copper, and wood were gathered using a spectrometer with advanced on-chip filtering [1]. This spectrometer, featuring a compact 9x9mm array of up to 8 wavelength-selective photodiodes, divides the spectrum into eight bands ranging from 400 to 1100 nm. The goal is to use ML/DL models to learn useful patterns for each material. This evaluation only considers 100 observations of Al- and Cu-diode. HesenbergBases shows the strongest performance overall, while other models like FNN, GRN, and RNN show minimal performance differences.

In Tab. 4, the performance of different models is compared using hypothesis tests on the *Al-diode* dataset. For FNN vs. HeisenbergBases, the p-value is 1.0000, showing no significant difference between the models. In the comparison of GRN vs. HeisenbergBases, the p-value is 0.99999, indicating no significant difference. However, in comparisons where HeisenbergBases is the better model—against FNN, GRN, RNN, and Transformer—the p-values are all smaller than 2.52×10^{-6}, signifying that the HeisenbergBases model significantly outperforms each of these models.

In Tab. 5, the performance of different models is compared using hypothesis tests on the *Cu-diode* dataset. HeisenbergBases is compared to FNN, GRN, RNN, and Transformer, the p-values are all very small, with values of 2.63×10^{-6}, 0.0004, 1.79×10^{-5}, and 1.79×10^{-5}, respec-

Index	Case Study	N_{train}	N_{test}	Ref.
1	Wigner function of Quantum Cat State	80	20	Simulated
2	Estimation of Temperature-Solubility Relation	144	37	BigSolBD [2]
3	Spectral Classifications of Materials (Al)	80	20	[1]
4	Spectral Classifications of Materials (Cu)	80	20	[1]

TABLE 1: Case studies with training and test data sizes

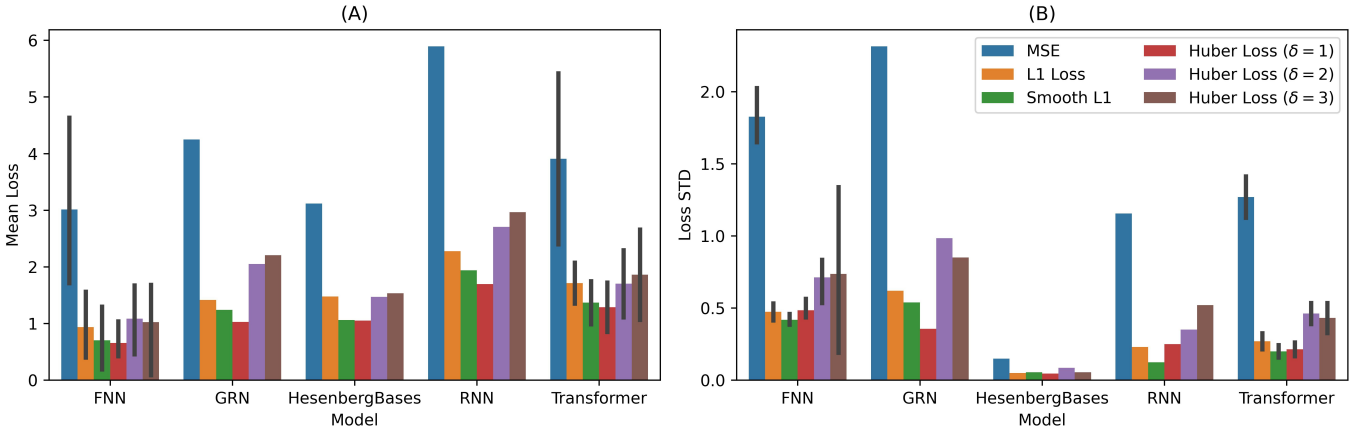

Fig. 4: Evaluation in quantum state learning task. (A) mean loss, (B) standard deviation of loss.

Hypothesis Test	p-value
FNN < GRN	0.0518
FNN < HesenbergBases	0.1398
FNN < RNN	0.0007*
FNN < Transformer	0.0146*
GRN < FNN	0.9552
GRN < HesenbergBases	0.6504
GRN < RNN	0.0898
GRN < Transformer	0.5542
HesenbergBases < FNN	0.8746
HesenbergBases < GRN	0.4091
HesenbergBases < RNN	0.0206*
HesenbergBases < Transformer	0.3082
RNN < FNN	0.9995
RNN < GRN	0.9340
RNN < HesenbergBases	0.9870
RNN < Transformer	0.9585
Transformer < FNN	0.9869
Transformer < GRN	0.4818
Transformer < HesenbergBases	0.7234
Transformer < RNN	0.0512

TABLE 2: P-value of statistical test among model's average performance in **quantum cat state** learning task

Hypothesis Test	p-value
FNN < GRN	0.0232*
FNN < HesenbergBases	0.9995
FNN < RNN	0.0043*
FNN < Transformer	0.0160*
GRN < FNN	0.9798
GRN < HesenbergBases	0.9999
GRN < RNN	0.0705
GRN < Transformer	0.1248
HesenbergBases < FNN	0.0007*
HesenbergBases < GRN	1.79×10^{-5}*
HesenbergBases < RNN	1.79×10^{-5}*
HesenbergBases < Transformer	0.0004*
RNN < FNN	0.9964
RNN < GRN	0.9370
RNN < HesenbergBases	0.9999
RNN < Transformer	0.8546
Transformer < FNN	0.9878
Transformer < GRN	0.8936
Transformer < HesenbergBases	0.9997
Transformer < RNN	0.1677

TABLE 3: P-value of statistical test among model's average performance in **BigSol** dataset

tively, indicating that HeisenbergBases significantly outperforms all these models.

Overal, the HeisenbergBases model shows significantly better performance than the FNN, GRN, RNN, and Transformer models, as indicated by the very small p-values in these comparisons. Meanwhile, the FNN, GRN, RNN, and Transformer models show no statistically significant differences in performance when compared with each other.

3.4 Cell Automata

In this section, we will demonstrate that the derived Heisenberg bases can be used as the cell decay rate in cell automata. From Fig. 3, we observe that the HeisenbergBases can represent the Black body radiation, which describes the electromagnetic radiation emitted by a perfect black body. It is an idealized physical object that absorbs all incident radiation and emits energy based on temperature. Specifically, associate u as the wavelength and $\Delta_x \Delta_p(t, \alpha, u)$ as the spectral radiance. We find that the radiance decays as we increase wavelength. We will generally use the Heisenberg bases as the decay rates for the cell's viability.

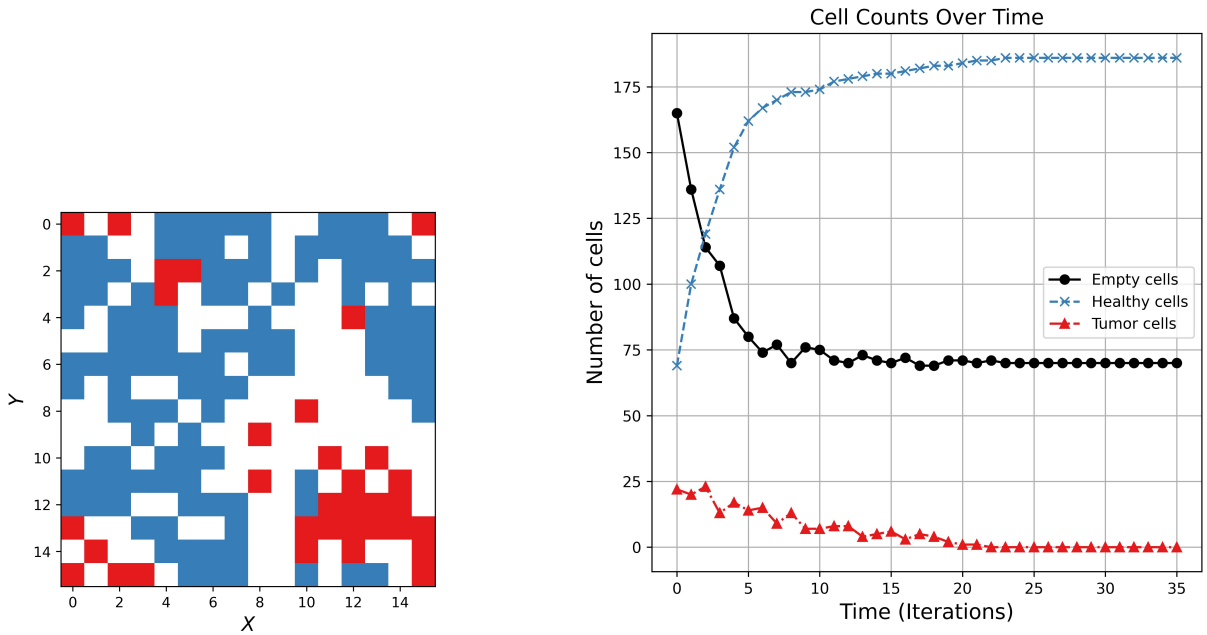

Fig. 5: Cell automata model of tumor-healthy cells competition. Left: spatial distribution of tumor (red), healthy (blue), and empty cells (white). Right: cell counts over time

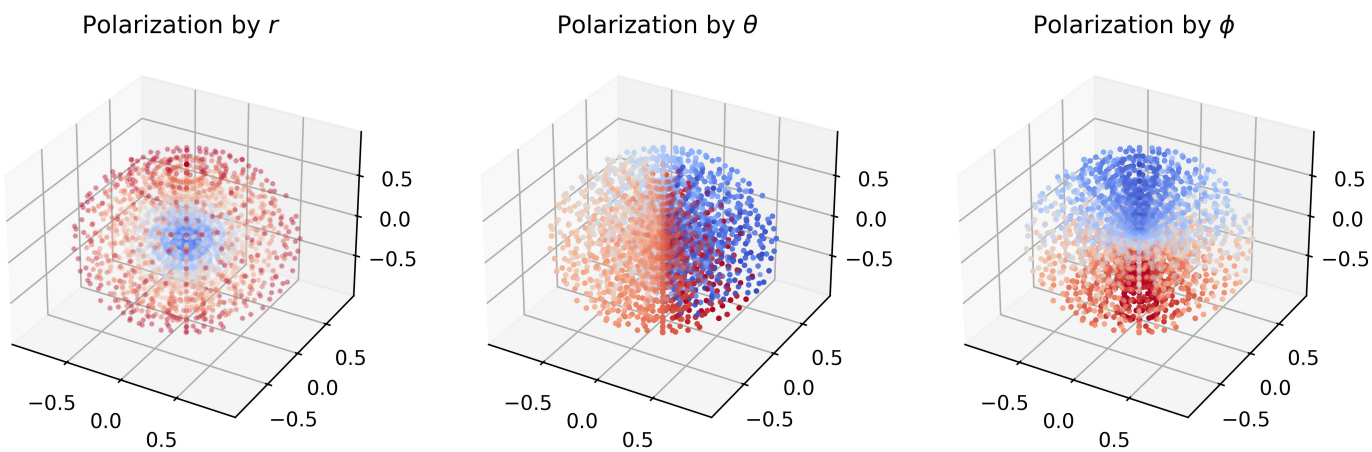

Fig. 6: Simulation of hydrogen's electron orbit

3.4.1 *Dynamical Simulation for Two Competitive Creatures*

This model simulates the evolution of a grid-based, 2D system in which cells can be in one of three states: Healthy, Tumor, or Empty. The Simulation proceeds over a specified number of iterations, with each cell's state updated based on its current state and the states of its neighboring cells. The model includes parameters to control the behavior of cell transitions, including decay and proliferation rates.

Inputs

- **Grid Size (M)**: Defines the dimensions of the grid, which is $M \times M$.
- **Number of Iterations (iter)**: Specifies the number of time steps the Simulation will run.
- **Time for Simulation (t)**: Represents the total simulation time, although not directly used in the current model description.

Model Dynamics For each iteration in the range iter:

1) **Grid Update**: The grid is updated using a function `update(grid)` to generate `new_grid`, which represents the state of the system after applying the transition rules.
2) **Computation of Factors**:
   - **Decay Factor (δ_d)**: Computed as $\delta_d = \frac{1}{N_b}\Lambda$, where N_b is the number of neighboring cells considered, and Λ is a parameter related to the decay rate (Equation 29).
   - **Proliferation Factor (δ_p)**: Defined as $\delta_p = \frac{1}{\delta_d}$, representing the rate of cell proliferation.
3) **Cell State Transition Rules**:
   - **Healthy Cell (grid[i, j] == 1)**:
     - With a small probability (0.01) and if there are existing tumor cells, the healthy cell may transition to a tumor cell.
     - Otherwise, the cell remains healthy.
   - **Tumor Cell (grid[i, j] == 2)**:

Hypothesis Test	p-value
FNN < GRN	0.3916
FNN < HesenbergBases	1.0000
FNN < RNN	0.0328*
FNN < Transformer	0.0328*
GRN < FNN	0.6246
GRN < HesenbergBases	0.9999
GRN < RNN	0.0705
GRN < Transformer	0.0705
HesenbergBases < FNN	1.55×10^{-7}*
HesenbergBases < GRN	2.52×10^{-6}*
HesenbergBases < RNN	2.52×10^{-6}*
HesenbergBases < Transformer	2.52×10^{-6}*
RNN < FNN	0.9702
RNN < GRN	0.9370
RNN < HesenbergBases	0.9999
RNN < Transformer	0.9214
Transformer < FNN	0.9702
Transformer < GRN	0.9370
Transformer < HesenbergBases	0.9999
Transformer < RNN	0.0874

TABLE 4: P-value of statistical test among model's average performance in **Al-diode** dataset

Hypothesis Test	p-value
FNN < GRN	0.1552
FNN < HesenbergBases	0.9999
FNN < RNN	0.0270*
FNN < Transformer	0.0270*
GRN < FNN	0.8602
GRN < HesenbergBases	0.9997
GRN < RNN	0.1454
GRN < Transformer	0.1454
HesenbergBases < FNN	2.63×10^{-6}*
HesenbergBases < GRN	0.0004*
HesenbergBases < RNN	1.79×10^{-5}*
HesenbergBases < Transformer	1.79×10^{-5}*
RNN < FNN	0.9755
RNN < GRN	0.8752
RNN < HesenbergBases	0.99999
RNN < Transformer	0.9214
Transformer < FNN	0.9755
Transformer < GRN	0.8752
Transformer < HesenbergBases	0.99999
Transformer < RNN	0.0874

TABLE 5: P-value of statistical test among model's average performance in **Cu-diode** dataset

  - The tumor cell may decay (transition to space) with a probability defined by `decay_factor`.
  - If there are fewer than three healthy cells and fewer than or equal to 3 tumor cells, the tumor cell remains a tumor cell.
  - Otherwise, the cell transitions to a space.
- **Empty Space (grid[i, j] == 0)**:
  - The space may become a tumor cell if there are exactly three tumor cells and a random number exceeds `decay_factor`.
  - Alternatively, it may become a healthy cell if there are exactly three healthy cells and a random number exceeds `proliferation_rate`.

Using our proposed cell automata model, we can simulate the spatial distribution of tumor, healthy cells, and space, shown in Fig. 5. Also, this Simulation suggests the dominance of healthy cells, and the number of empty cells converges at the end of the Simulation.

3.4.2 Simulation of Hydrogen's Electron Orbit

We design an automata model for a 3D model, which simulates the electron's position in the Hydrogen atom. First, we create a polar grid of (r, θ, ϕ) coordinate. The cell is active when the electron is present; otherwise, it is inactive. We show the simulation result in Fig. 6, colored by the value r, θ, ϕ. The low values are in blue, and the high ones are in red. The update rule is given as if a random value ¡ decay rate, `newcells`[i, j, k] = 1 - `cells`[i, j, k] (Flip state (0 to 1 or 1 to 0)). In Fig. 6, the electron with high values of r is in the outer, while the smaller values are closer to the proton. Besides, we can use vertical and horizontal surfaces to stratify the position by θ and ϕ.

4 Discussion

4.1 The Link to the Heisenberg Uncertainty

Back to Equa. 4, Δ_x and Δ_p are the second moment $\mathbb{E}(x^2)$ and $\mathbb{E}(p^2)$, respectively. We have

$$\begin{aligned} \sigma_X^2 &= \mathbb{E}(X^2) - \mathbb{E}(X)^2 = \Delta_x - \mu_x^2 \\ \sigma_P^2 &= \mathbb{E}(P^2) - \mathbb{E}(P)^2 = \Delta_p - \mu_p^2 \end{aligned} \tag{34}$$

Then,

$$\sigma_X^2 \sigma_P^2 = \Delta_x \Delta_p + (\mu_x \mu_p)^2 - \Delta_x \mu_p^2 - \Delta_p \mu_x^2 \tag{35}$$

Since both Δ_x and Δ_p is assumed to be positive, we have

$$\sigma_X^2 \sigma_P^2 \geq \Delta_x \Delta_p + (\mu_x \mu_p)^2 \geq \Delta_x \Delta_p \tag{36}$$

4.2 Difference from Classical Neural Networks

In classic, Each neuron in a layer computes a weighted sum of its inputs, applies a bias, and then passes the result through a nonlinear activation function. For a neuron j in layer l:

$$z_j^{(l)} = \sum_i w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}$$

where $z_j^{(l)}$ is the weighted sum input to neuron j in layer l; $w_{ji}^{(l)}$ is the weight connecting neuron i from layer $l-1$ to neuron j in layer l; $a_i^{(l-1)}$ is the activation (output) of neuron i in layer $l-1$; $b_j^{(l)}$ is the bias for neuron j in layer l. The classical embedding is linear embeddings. In contrast, our embedding in Equa. 30 is non-linear due to Equa. 28, showed in Fig. 3.

4.3 Future Works

4.3.1 Scalability of the model

Adding more generative functions α results in more complex Heisenberg bases at the expense of higher computation time. This could lead to a better model as we provide a more general class of random functions. Parallel computing can reduce execution time and mitigate the increased computational cost.

4.3.2 Further applications

The proposed models can be bench-marked in other learning tasks and datasets. Additionally, the cellular automata can be applied to practical applications in biomedical science, physics, chemistry, and machine intelligence. Furthermore, the stochastic electron density models can be utilized for problems in atom and molecular dynamics or quantum chemistry.

4.3.3 Limitations

The proposed model's universal approximation capability remains uncertain and warrants further investigation in future work. The simulation time for the cellular automata is notably high, necessitating methods to reduce computational costs. We view this work as a preliminary effort to bridge quantum theory with modern machine intelligence. Future research should focus on exploring more intriguing quantum phenomena, such as quantum tunneling and quantum entanglement.

5 Conclusion

To this point, we have introduced a framework to construct AI models based on the parameterized uncertainty (Section 2.4). The uncertainty function is derived from our geometric interpretation in Section 2.1. Three use cases of our proposed framework are demonstrated, which concern applications in quantum computing, material science, and cell automata modeling (Section 3). Some insights into the proposal and its future extensions are also discussed in the article.

References

[1] T. Bhatt, R. Soni, M. Upadhyay, H. Jayswal, J. Chaudhari, N. Dubey, A. Patel, S. Sharma, and A. Makwana. A spectral dataset of different materials for metal classification, 2024.

[2] L. Krasnov, S. Mikhaylov, M. Fedorov, and S. Sosnin. Bigsoldb: Solubility dataset of compounds in organic solvents and water in a wide range of temperatures. 2023.

[3] I. Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[4] N. Nguyen and K.-C. Chen. Bayesian quantum neural networks. *IEEE Access*, 10:54110–54122, 2022.

[5] N. Nguyen and K.-C. Chen. Quantum embedding search for quantum machine learning. *IEEE Access*, 10:41444–41456, 2022.

[6] P.-N. Nguyen. The duality game: a quantum algorithm for body dynamics modeling. *Quantum Information Processing*, 23(1):21, 2024.

[7] P.-N. Nguyen. Quantum dna encoder: A case-study in grna analysis. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 232–239. IEEE, 2024.

[8] P.-N. Nguyen. Quantum word embedding for machine learning. *Physica Scripta*, 99(8):086004, 2024.

6 Appendix

Visualization of Wigner functions for Generating Quantum Coherent States

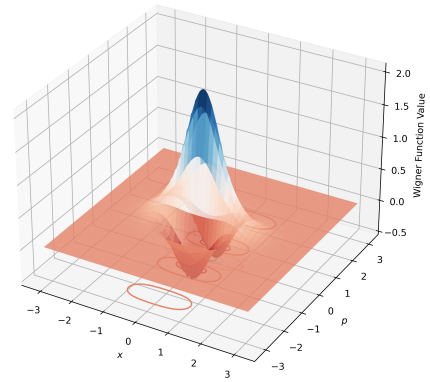

Fig. 7: Generating Wigner functions of Quantified Quantum Coherent States

Distribution Name	Mathematical Equation
Uniform	$f(x) = 1,\ 0 \leq x \leq 1$
Normal	$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$
Binomial	$f(x) = \binom{n}{x} p^x (1-p)^{n-x}$
Poisson	$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}$
Exponential	$f(x) = \frac{1}{\text{scale}} e^{-x/\text{scale}}$
Geometric	$f(x) = p(1-p)^x$
Negative Binomial	$f(x) = \binom{n+x-1}{x} p^n (1-p)^x$
Beta	$f(x) = \frac{x^{a-1}(1-x)^{b-1}}{\mathrm{B}(a,b)}$
Gamma	$f(x) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-x/\theta}$
Chi-squared	$f(x) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{(k/2)-1} e^{-x/2}$
Student's t	$f(x) = \frac{\Gamma\left(\frac{\text{df}+1}{2}\right)}{\sqrt{\text{df}\pi}\,\Gamma\left(\frac{\text{df}}{2}\right)} \left(1 + \frac{x^2}{\text{df}}\right)^{-\frac{\text{df}+1}{2}}$
Weibull	$f(x) = c x^{c-1} e^{-x^c}$
Lognormal	$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln x - \mu)^2/(2\sigma^2)}$
Cauchy	$f(x) = \frac{1}{\pi \text{scale}\left(1 + \left(\frac{x - \text{loc}}{\text{scale}}\right)^2\right)}$
Bernoulli	$f(x) = p^x (1-p)^{1-x}$
Hypergeometric	$f(x) = \frac{\binom{n_{\text{good}}}{x}\binom{n_{\text{bad}}}{n_{\text{sample}}-x}}{\binom{n_{\text{good}}+n_{\text{bad}}}{n_{\text{sample}}}}$
Discrete Uniform	$f(x) = \frac{1}{\text{high} - \text{low} + 1}$
Triangular	$f(x) = \begin{cases} \frac{2(x - \text{left})}{(\text{right} - \text{left})(\text{mode} - \text{left})}, & \text{if left} \leq x \leq \text{mode}, \\ \frac{2(\text{right} - x)}{(\text{right} - \text{left})(\text{right} - \text{mode})}, & \text{if mode} \leq x \leq \text{right}. \end{cases}$
Rayleigh	$f(x) = \frac{x}{\text{scale}^2} e^{-x^2/(2\text{scale}^2)}$

TABLE 6: The generating distribution $\alpha(t)$ used in our HeisenbergBases