Quantum-StateHD: Leveraging Random Quantum States for Hyperdimensional Learning

Anonymous Authors

***Abstract*—Hyperdimensional Computing (HDC) is a brain-inspired computational paradigm that utilizes high-dimensional vector spaces to represent, bind, and process information efficiently. Its robustness, scalability, and adaptability make HDC particularly effective for machine learning tasks such as classification and pattern recognition. In this work, we introduce Quantum-StateHD (QSHD), a novel approach that leverages random quantum states as encoding mechanisms for high-dimensional spaces within the HDC framework. Quantum states, with their inherent properties of high-dimensionality, near-orthogonality, and probabilistic distributions, provide a natural substrate for constructing robust representations. Quantum-StateHD encodes classical data into the coefficients of random quantum states and seamlessly integrates these representations into the HDC pipeline. Experimental results on several benchmark classification datasets demonstrate that Quantum-StateHD achieves competitive or superior performance compared to traditional state-of-the-art HDC methods, showcasing its potential for robust and efficient high-dimensional information processing. This study bridges the gap between quantum information science and HDC, offering a promising direction for hybrid quantum-classical machine learning architectures.**

***Index Terms*—Quantum State, Hyperdimensional Computing, Quantum Encoding.**

I. Introduction

Hyperdimensional Computing (HDC) is an emerging computation paradigm inspired by the human brain processing information using distributed and hyperdimensional representations. Its inherent properties, such as small model size, low computational complexity, and support for one-shot and few-shot learning, make HDC well-suited for resource-constrained environments. HDC is foreseen as a valuable tool in applications where efficiency is prioritized over absolute accuracy. It shows performance in many applications, including signal processing [1], robotics [2], natural language processing [3], and data security [4].

HDC can be conceptualized as a computational approach that employs high-dimensional vectors to represent information. These vectors, typically with thousands of elements, enable efficient processing through vector operations. Various encoding approaches are proposed to map objects in the input space to the hyperspace, and the choice of encoding functions depends on the characteristics of the input data. For example, binary encoding represents data as binary strings, which are then mapped to high-dimensional vectors. N-grams decompose sequential data, such as text or time series, into overlapping subsequences of length n. Each n-gram is then encoded as a vector. These encoding strategies share a common characteristic: they begin by encoding elementary data units, such as characters, pixel intensities, or signal amplitudes, which are subsequently aggregated to represent composite objects (e.g., images, sentences, and signals).

Although encoding is crucial for representing data within the HDC framework, existing methods face inherent limitations, particularly with respect to computational cost and potential information loss [5]. Generation and manipulation of high-dimensional vectors, especially for complex data types, can require substantial computational resources, posing a scalability challenge for large datasets [6]. Furthermore, the encoding process itself may lead to a reduction in the fidelity of the data representation. This loss of information can be due to dimensionality reduction techniques or inherent constraints of the chosen encoding scheme, which can affect the accuracy of subsequent HDC operations [7]. Therefore, balancing representational capacity with computational efficiency and minimizing information loss remain key considerations in the development of effective HDC encoding strategies.

To address the above issues, this work presents Quantum-StateHD (QSHD), a novel framework designed to enhance encoding efficiency within the HDC paradigm by exploiting the principles of quantum mechanics. Quantum-StateHD leverages quantum encoding, a technique used in quantum computing and communication where information is represented through quantum states. A pure state is the simplest kind of quantum state, which can be described by a normalized state vector, $\psi = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers that satisfy $|\alpha|^2 + |\beta|^2 = 1$. This representation describes the state of a qubit, the quantum analog of a classical bit. A key property of qubits is their ability to exist in a superposition of states, which means that they can simultaneously embody a combination of $|0\rangle$ and $|1\rangle$. This characteristic allows n qubits to span a Hilbert space of dimension 2^n. Consequently, the information capacity of a quantum system scales exponentially with the number of qubits, offering a substantial advantage in information storage compared to classical systems where capacity increases linearly. The proposed QSHD utilizes this exponential scaling to achieve more efficient encoding within the HDC framework.

Integrating quantum encoding into the HDC paradigm is non-trivial. The primary challenge is how to utilize quantum mechanisms for data encoding. Common quantum encoding methods include binary encoding and amplitude encoding. To address this challenge, we adopt a combined strategy that integrates the real and imaginary parts along with the amplitude encoding. The second challenge involves an approximate computation of orthogonality and similarity. In high-

dimensional spaces, approximate orthogonality is a crucial property of hyperdimensional computing (HDC), which ensures robustness in noisy environments. Therefore, proving the approximate orthogonality and similarity computation of the encoded random quantum states in high-dimensional Hilbert spaces presents a significant challenge. The final challenge is computational complexity. On classical computing platforms, HDC implementations typically have low complexity, which is one of the main reasons why HDC is widely chosen for real-time tasks such as edge computing and embedded AI. Thus, validating the efficiency of our proposed method on benchmark datasets is a practical challenge. To address the aforementioned challenges, the main contributions of our proposed QSHD method are as follows:

- **Proposing a novel encoding method using quantum states:** We present a method that utilizes random quantum states as the foundation for encoding data into hyperdimensional spaces. Quantum states, with their inherent high-dimensionality and probabilistic distributions, provide a natural substrate for representing complex data. This encoding mechanism replaces traditional deterministic or classical random hypervector schemes, offering improved representational capacity by exploiting the unique properties of quantum states, such as near-orthogonality and stochasticity. The method efficiently maps classical data into quantum-state-based representations, preserving the underlying structure of the data while enhancing robustness.
- **Theoretical Analysis of Quantum-State Properties in HDC:** We provide a rigorous theoretical analysis that demonstrates how quantum states align with the requirements of HDC. This includes proving that the near-orthogonality of random quantum states enhances the separability of encoded representations, ensuring minimal interference between distinct data points. This theoretical foundation establishes the feasibility and effectiveness of quantum-state-based encoding within HDC.
- **Extensive Evaluation on Benchmark Datasets:** The performance of Quantum-StateHD was evaluated on multiple benchmark datasets commonly used in classification tasks. To validate the practical implementation, we implemented our method using standard Python libraries and also leveraged IBM's Qiskit framework for quantum computing simulations. The experimental results demonstrate that Quantum-StateHD achieves competitive or superior performance compared to traditional HDC methods in terms of accuracy, robustness, and scalability. These findings highlight the potential benefits of integrating quantum-inspired encoding strategies into classical machine learning pipelines.

The remaining paper is organized as follows. Section II presents related works. Section III introduces the traditional HD methods. Section IV provides the overview of the proposed approach and detailed designs of each component. Section V and VI comprehensively evaluate the proposed approach and compare the performance with the state-of-the-art baselines. Finally, section VII concludes the paper.

II. Related Work

Inspired by the neural activity patterns of the human brain, hyperdimensional computing (HDC) has emerged as a promising computing framework. The basic HDC model [8] establishes the fundamental principle of representing data with high-dimensional random vectors. Significant research efforts focus on improving the efficiency of the HDC model, encoding strategies, and hardware implementation. Wang proposes DistHD [9], a learner-aware dynamic encoding method that achieves a significant reduction in dimension while improving inference speed. Similarly, Imani et al. develop QuantHD [10], an efficient HDC framework that reduces reliance on floating-point numbers, thus greatly improving energy efficiency. Duan et al. propose LeHDC [11], which enhances learning-based HDC classifiers through binary neural network training and improves the accuracy of inference. Training efficiency is also a key area of focus. AdaptHD [12] is an adaptive training method to optimize HDC learning, achieve faster training, and achieve significant energy savings. OnlineHD [13] supports single-shot online learning and enhances the robustness to hardware failures. Several works address resource efficiency and hardware adaptability. Morris et al. introduce CompHD [14], focusing on model compression to reduce storage costs while improving execution speed. SparseHD [15], which uses sparse-aware computing for hardware acceleration, resulting in lower resource consumption. To support real-time edge computing, Zou et al. propose NeuralHD [16], a neural adaptation-inspired dynamic coding system that can scale HDC across IoT systems with faster training performance. These advances in HDC illustrate the collaborative efforts to develop scalable, efficient, and accurate models. Through innovations in coding schemes, adaptive learning, hardware-aware design, and edge computing, HDC has solidified its role as a powerful framework for future computing paradigms, especially in energy-constrained environments.

Quantum encoding is a critical part of quantum computing, translating classical data into quantum states through encoding techniques such as amplitude encoding, basis encoding, and hybrid methods. These encoding schemes facilitate quantum computation, enhance data security, and enable efficient quantum communication. Ranga et al. [17] highlighted the transformative role of data encoding in quantum machine learning, focusing on improving model accuracy through optimized quantum state preparation. Similarly, Rath and Date [18] explored classical-to-quantum mapping, demonstrating its impact on the performance of quantum algorithms. Weigold et al. [19] expanded quantum data encoding patterns for gate-based quantum computers, while Bravo-Prieto [20] developed enhanced quantum autoencoders for data compression. In the field of quantum cryptography, Yang et al. [21] proposed a novel encryption scheme using quantum Fourier transforms for secure image communication. Image encoding applications were advanced by Yetiş and Karaköse [22], who design

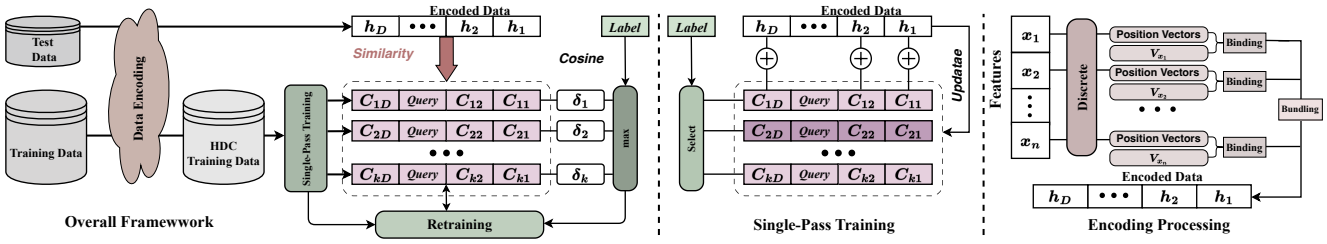

Fig. 1. Traditional HDC for Classification framework.

cost-effective quantum circuits. Hybrid quantum encoding techniques, explored by Smanchat et al. [23], demonstrate improved quantum data storage and processing. Furthermore, Majji et al. [24] introduce a quantum-based approach for image data compression, enhancing efficiency in quantum sensing. Nguyen et al. (2024) revisit the encoding in quantum neural networks, optimizing visual feature extraction for computer vision tasks. Recently, Govorov et al. [25] explore time-encoded photonic quantum states for applications in secure quantum communication protocols like quantum key distribution (QKD). These advancements showcase quantum encoding's versatility in enabling next-generation computing, data security, and communication technologies.

III. Background

A. Hyperdimensional Computing

Hyperdimensional computing can be applied to a variety of learning problems. This paper focuses on classification tasks and common problems in supervised learning. As shown in Fig. 1, applying hyperdimensional computing for classification tasks involves encoding, single-pass training, and similarity checking.

1) Encoding: Encoding is a key step in HDC to map input data into a high-dimensional vector space. The features of the input data and its structural relationships are embedded into a single high-dimension vector while retaining its information. Let $\mathbf{x} = [x_1, x_2, ..., x_n]$ denote preprocessed input data. Encoding $\mathbf{x}$ into a high-dimensional vector space usually takes the following steps.

Value Encoding: Value encoding maps each feature in the input data to a high-dimensional space. For discrete features, each possible value v_j is assigned a unique hypervector. If the features are continuous, such that x_i is within a range$[x^{low}, x^{high}]$, the value vector is computed as $\mathbf{V}_{x_i} = \alpha \cdot \mathbf{V}_{x^{low}} + (1-\alpha) \cdot \mathbf{V}_{x^{high}}$, where $(\alpha = x_i - x^{low})/(x^{high} - x^{low})$. This ensures that continuous values are smoothly encoded into high-dimensional space. Alternatively, if the value is discretized into binary or hot vectors, such as [1, 1, 1, 0, 0, 0], thermometer encoding can be applied to represent the feature's relative position with its range.

Position Encoding: Position encoding is a technique used to represent the position of a feature in a sequence. This is done by adding a vector, $\mathbf{P}_i \in \mathbb{R}^D$, where D is the dimension of the high-dimensional space, to the feature's representation that encodes its position. There are many different ways to do position encoding, but one common approach is to use a set of sinusoidal functions with different frequencies. Position encoding can be used to improve the performance of HDC models on tasks that involve sequential data, such as natural language processing.

Based on the above encoding method, a unique property of hyperdimensional space is a large number of nearly orthogonal hypervectors, enabling highly parallel operations. We can use these operations to combine the positional information $\mathbf{P}_i$ with the value information $\mathbf{V}_{x_i}$ for each feature, creating a bound vector $\mathbf{B}_i$. This is achieved by mathematical operations **Binding**. Binding operation can be performed using various methods, such as dot product, $\mathbf{B}_i = \mathbf{P}_i \cdot \mathbf{V}_{x_i}$, element-wise multiplication, $\mathbf{B}_i = \mathbf{P}_i \odot \mathbf{V}_{xi}$, or XOR for binary vectors, $\mathbf{B}_i = \mathbf{P}_i \oplus \mathbf{V}_{x_i}$. The result of this operation ensures that both position and value information are encoded into a single vector for each feature. Once all features have been encoded as bound vectors, bundling is performed to combine them into a hypervector $\mathbf{H}$ that represent the entire input data. The **bundling** operation aims to superimpose the bund vectors, typically by summing them: $\mathbf{H} = \sum_{i=1}^{n} \mathbf{B}_i$, where n is the total number of features. Optionally, weights ω_i can be assigned to the features during binding, resulting in a weighted sum: $\mathbf{H} = \sum_{i=1}^{n} \omega_i \cdot \mathbf{B}_i$. This superposition ensures that all feature information is integrated into a single vector, capturing both individual feature contributions and their relationships. Finally, the hypervector $\mathbf{H}$ is normalized to ensure numerical stability for further processing. Normally, L2 normalization is applied.

2) Training: In the HDC classification task, most existing HDC algorithms perform the training in a single iteration. It can be divided into several steps. At the beginning of training, each class $\mathbf{C}_k$ is associated with a class hypervector $\mathbf{C}_k \in \mathbb{R}^D$, where D is the dimensionality of the hypervector space. These class hypervectors are initialized as $\mathbf{C}_k = 0, \forall k \in 1, 2, ..., K$, where K is the number of classes. This initialization allows the hypervectors to accumulate information from the samples during training. Then, for every training sample $(\mathbf{x}, y)$, where $\mathbf{x}$ represents the features and y is the class label, the encoded normalized hypervector of the sample, $\mathbf{H}$ is added to the corresponding class hypervector: $\mathbf{C}_k = \mathbf{C}$. After all samples have been processed, the class hypervectors C_k finish updating. Normally, each class hypervector should be

normalized to ensure consistency during inference. The class hypervectors represent the comprehensive feature information of their respective class and serve as reference vectors for classification during inference.

3) Inference: To classify the input sample, the similarity between the encoded sample hypervector $\mathbf{H_x}$ and each class hypervector $\mathbf{C}_k$ is calculated. There are many ways to calculate similarity. The most popular one is the cosine similarity that measures the cosine of the angle between two hypervectors: $\delta(\mathbf{H_x}, \mathbf{C}_k)$ where both $\mathbf{H_x}$ and $\mathbf{C}_k$ are normalized. If the hypervectors are binarized, the similarity can be computed as the complement of the Hamming distance: $d(\mathbf{H_x}, \mathbf{C}_k) = \sum_{i=1}^{D}(\mathbf{H_x}[i] \oplus \mathbf{C}_k[i])$, where $\oplus$ is also the XOR operation, and the smaller the distance, the higher the similarity. For the real-valued hypervectors, a simple dot product can also serve as a similarity metric. The predicted label is determined by selecting the class k with the highest similarity score:

$$\hat{y} = \arg\max_k \delta(\mathbf{H_x}, \mathbf{C}_k)$$

B. Quantum state and quantum encoding

1) Qubits and quantum states: A **qubit** is the fundamental unit of quantum information, analogous to a classical bit but with significant differences. Unlike a classical bit, which can only be in one of two states (0 or 1), a qubit can exist in a superposition of states 0 and 1. The state of a qubit is represented as: $\psi = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers that satisfy $|\alpha|^2 + |\beta|^2 = 1$. In addition, Assuming an n-qubit space exists, the state of a quantum system is represented as a vector in a 2^n-dimensional Hibert space. It captures all possible combinations of the n qubits in superposition. A general **quantum state** for n-qubits can be written as

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle$$

where $|\cdot\rangle$ represents Dirac notation, $|i\rangle$ represent the basis vector like $|0...000\rangle$,$|0...001\rangle$, or $|0...010\rangle$, $c_i \in \mathbb{C}$ are the coefficients corresponding to each basis state, satisfying the normalization condition:

$$\sum_{i=0}^{2^n-1} {c_i}^2 = 1$$

which also matches the definition of a pure state.

2) Quantum encoding: Quantum encoding is the process of mapping classical information into quantum states for storing, processing, or transmission. One of the efficient, simple, and pure-state methods is amplitude coding. Consider a given classical data vector $\mathbf{x} = [x_0, x_1, ..., x_{N-1}]$, where $N = 2^n$ and n is the number of qubits, amplitude encoding transforms this vector into a quantum state $|\psi\rangle$ as follows:

$$|\psi\rangle = \frac{1}{||\mathbf{x}||} \sum_{i=0}^{N-1} x_i |i\rangle$$

Based on such an encoding method and HD encoding process, we can prepare and characterize the random quantum pure states of the original data.

IV. Proposed Quantum-StateHD

The overall structure of our proposed Quantum-StateHD is shown in Fig. 2, which integrates quantum encoding and representation into the conventional HDC framework. The details of quantum encoding block, as well as the training and inference processes are detailed in the following sections.

A. Quantum-StateHD Encoding

This section presents the proposed encoding method inspired from the quantum mechanism and signal processing. By mapping the original feature to a high-dimensional quantum space, the nonlinear strengthening and numerical stability of the feature can be obtained simultaneously. The specific methods and instructions are as follows. Suppose $\mathbf{x} = [x_1, x_2, ..., x_n]$ is the input data with n features, which is required to be mapped to a hypervector $\mathbf{H} = \{h_1, h_2..., h_D\}$ with dimensions D, where $D = 2^N$ and $h_i \in \mathbb{C}$, and $h_i = Re(h_i) + iIm(h_i)$ is a complex number with $Re(h_i)$ and $Im(h_i)$ represent the real and imaginary parts, respectively. Firstly, two vectors $\mathbf{B}^{Re} = \{B_1^{Re}, B_2^{Re}, ...B_D^{Re}\}$ $\mathbf{B}^{Im} = \{B_1^{Im}, B_2^{Im}, ...B_D^{Im}\}$ are randomly generated from a Gaussian distribution with the mean $\mu = 0$ and the standard deviation $\sigma = 1$. We encode each element h_i of $\mathbf{H}$ by equation

$$Re(h_i) = cos(\mathbf{B}_i^{Re} \cdot \mathbf{x} + b_i^{Re}) \times sin(\mathbf{B}_i^{Re} \cdot \mathbf{x})$$
$$Im(h_i) = cos(\mathbf{B}_i^{Im} \cdot \mathbf{x} + b_i^{Im}) \times sin(\mathbf{B}_i^{Im} \cdot \mathbf{x})$$

where b_i^{Re} and b_i^{Im} are random shift value generated uniformly from $[0, 2\pi]$. This encoding method, based on random projection, trigonometric transformation, and random phase shifts, offers several key advantages.

Firstly, the proposed encoding method produces a complex hypervector $\mathbf{H}$ which is assumed to be normalized. This complex hypervector can directly serve as the coefficients of a pure quantum state composed of N-qubits. Each component h_i represents the complex amplitude of the corresponding quantum basis state $|i\rangle$ in the computational basis.

$$\mathbf{H} := |\psi\rangle = \sum_{i=0}^{2^N-1} h_i |i\rangle, h_i \in \mathbb{C}$$

This encoding process effectively transforms the original classical feature vector into a D-dimensional hypertvector, which can equivalently be viewed as a pure quantum state in an N-qubit Hibert space where $D = 2^N$.

Secondly, our method can also ensures to map different input vectors to nearly orthogonal quantum states even for large datasets. This kind of orthogonality is the reflection of the concentration of measure in classical high-dimensional space and also can extended to quantum space. Consider there are two random pure states:

$$|\psi\rangle = \sum_{i=0}^{2^N-1} c_i |i\rangle, \quad |\phi\rangle = \sum_{i=0}^{2^N-1} d_i |i\rangle,$$

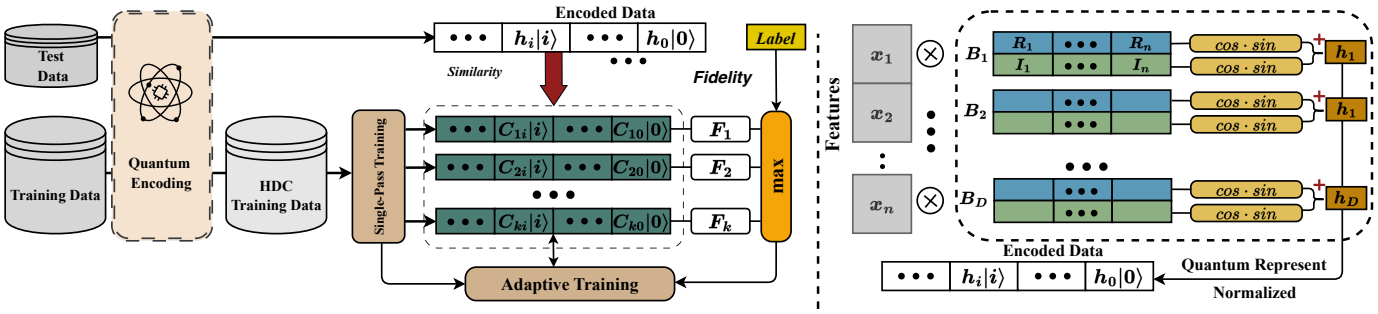

Fig. 2. Quantum-StateHD for Classification framework.

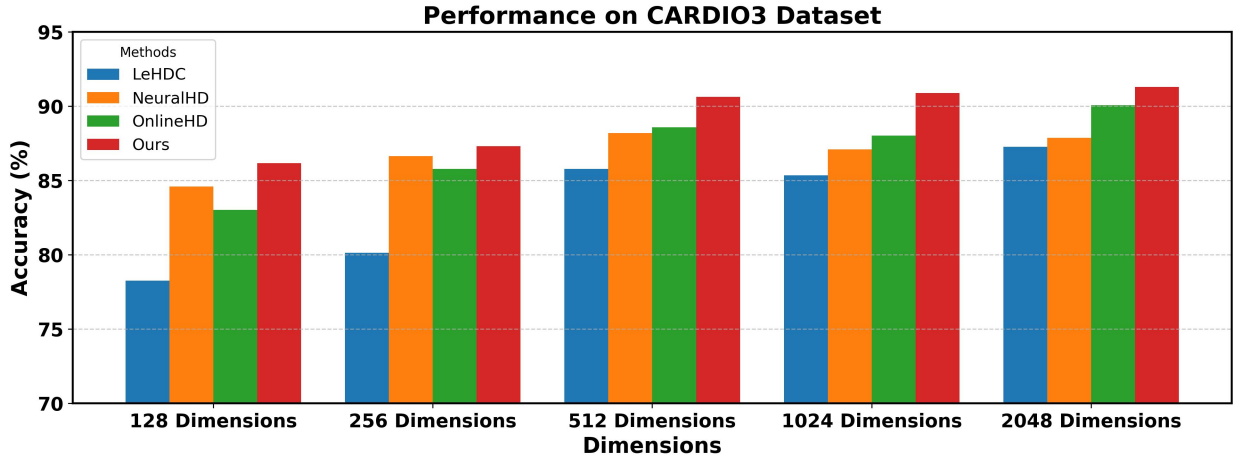

Fig. 3. Classification accuracy comparison between different methods

where $c_i, d_i \in \mathbb{C}$ are the normalized coefficients. In the quantum space, the orthogonality of two quantum states can be measured by **fidelity**. The fidelity of two pure state is defined as $F(|\psi\rangle, |\phi\rangle) = |\langle\psi|\,|\phi\rangle|^2$. The value approaching zero indicates that the orthogonality between two quantum states becomes stronger (if $F(\cdot) = 1$, the two states are identical, and if $F(\cdot) = 0$, the two states are orthogonal). The inner product between the states is:

$$\langle\psi|\phi\rangle = \sum_{i=0}^{2^N-1} c_i^* d_i$$

Since c_i and d_i are independent and symmetrically distributed, we have expectation value $\mathbf{E}[\langle\psi|\phi\rangle] = 0$. The variance should be the expectation of the squared magnitude of the inner product:

$$\mathbf{E}[|\,\langle\psi|\phi\rangle\,|^2] = \sum_{i=0}^{2^N-1} \mathbf{E}[|c_i|^2] \cdot \mathbf{E}[|d_i|^2]$$

Since $\mathbf{E}[|c_i|^2] = \mathbf{E}[|d_i|^2] = 1/2^N$, it follows that:

$$\mathbf{E}[|\,\langle\psi|\phi\rangle\,|^2] = \frac{1}{2^N}$$

Thus, fidelity $F(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|^2$ decreases as 2^N increases, and at the limit $2^N \to \infty$, the fidelity tends to zero. The fidelity calculation is also used as a measure of similarity in subsequent classification training and inference. This method makes different feature vectors encoded into high-dimensional quantum states distinguishable. Therefore, this encoding approach is well suited for application in hyper-dimensional computing.

B. Quantum-StateHD Training

Quantum-StateHD computing supports efficient one-pass training. The encoder maps all training data to the quantum state $|\psi\rangle$. Training $|\psi\rangle$ in the same class $\{|\psi_l^1\rangle, |\psi_l^2\rangle, ..., |\psi_l^m\rangle\}$ can be aggregated to generate a single class $|C_l\rangle$, where m means the number of samples in this class and l indicates the class tag. To ensure that the class superposition state is a pure state, we cannot add directly as in classical space.

$$|C_l\rangle = |\psi_l^1\rangle + |\psi_l^2\rangle, ..., |\psi_l^m\rangle$$

Assuming that each sample contributes equally to this class, the average superposition of quantum states should be used to obtain a pure state to express the class superposition state. The method is as follows:

$$|C_l\rangle = \frac{\sum_{i=1}^m |\psi_l^i\rangle}{\left\|\sum_{i=1}^m |\psi_l^i\rangle\right\|} = \frac{\sum_{i=1}^m |\psi_l^i\rangle}{\sqrt{m + \sum_{i\neq j} \langle\psi_l^i|\psi_l^j\rangle}}$$

In the quantum encoding section, it is explained that $|\psi_l^i\rangle$ and $|\psi_l^j\rangle$ $(i \neq j)$ are approximately orthogonal, so

$$\sum_{i \neq j} \langle \psi_l^i | \psi_l^j \rangle \approx 0$$

To implement this on a classical computer, the above mathematical operations need to be performed. However, on quantum devices or quantum platforms, multi-qubits unitary operations must be used to achieve the desired effect. For different classes, we use this method to obtain the superposition state for each class, allowing the classification of test data. Moreover, the iterative learning process is conducted as described in Algorithm 1. The QSHD incremental learning algorithm efficiently updates the model by building on previously learned representations instead of starting from scratch. In each training iteration, the algorithm processes each data point by calculating its similarity to the current model and predicting its category. If the prediction is correct, the model remains unchanged. However, if a misclassification occurs, the model is updated by adding the data point to the correct class representation and subtracting it from the incorrectly predicted class. This selective update mechanism allows the model to improve its performance while retaining prior knowledge, achieving fast convergence and continuous learning through multiple iterations.

Algorithm 1 Quantum-StateHD Incremental Learning

Input: Dataset $D = \{(x_i, y_i)\}$, Current Model M, Number of Iterations T
Output: Updated Model M
for $t = 1$ **to** T **do**
  for each $(x_i, y_i) \in D$ **do**
    Compute similarity S of x_i with the current model M
    $\hat{y}_i \leftarrow \arg\max S$ ▷ Predicted class for x_i
    if $\hat{y}_i == y_i$ **then**
      continue ▷ No update if correctly classified
    else
      $M[y_i] \leftarrow M[y_i] + x_i$ ▷ Add to correct class
      $M[\hat{y}_i] \leftarrow M[\hat{y}_i] - x_i$ ▷ Subtract from incorrect class
    end if
  end for
end for
return M

C. Quantum-StateHD Inference

The inference process is similar to the classical HD classification method. A sample of test data $\mathbf{x_t}$ should be encoded into a quantum state $|\psi_t\rangle$. The similarity metric fidelity is used to measure the strength of a match between $|\psi_t\rangle$ and each class $|C_l\rangle$. After the fidelity is computed, the class corresponding to the highest fidelity value is selected as the output.

V. Experimental Setup

We evaluate the proposed method by conducting comparative experiments on various datasets and dimensions, comparing it against state-of-the-art baseline methods to obtain detailed experimental results and parameter information. Specific details are provided below.

A. Datasets

We test our methods and state-of-the-art methods on 5 benchmark classification datasets.

CARDIO3 and CARDIO10: CARDIO [26] is a dataset designed for in-depth research on cardiac signals. It provides tools to load and process ECG signals and includes predefined workflows for training models and performing inference. This dataset is often utilized for detecting PQ, QT, QRS segments, and calculating heart rate.

MNIST: The MNIST dataset [27] is a reference for handwritten digit recognition tasks. It contains grayscale images of digits ranging from 0 to 9 and is widely used in machine learning and deep learning research to evaluate image classification algorithms.

ISOLET: The ISOLET dataset [28] is designed for speech recognition tasks, specifically to classify spoken letters of the English alphabet. It provides acoustic data collected from multiple speakers and is commonly used to develop and test audio classification models.

UCIHAR: The UCI Human Activity Recognition (UCIHAR) [29] dataset focuses on identifying human activities using data collected from wearable sensors. It has been widely used in machine learning to study time series data and activity recognition applications.

The quantitative information of these data is shown in Table II.

B. Baselines

A variety of HD baselines in multiple dimensions are used as the baselines in our experiments. Specifically, since our method operates in the Hilbert space, all high-dimensional settings are chosen as powers of 2 for a fair comparison. The baselines used include Vanilla HD, AdaptHD, DistHD, CompHD, NeuralHD, SparseHD, QuantHD, LeHDC, and OnlineHD, with dimensions $\{128, 256, 512, 1024, 2048, 4096, 8192\}$. These correspond to our method operating with $\{7, 8, 9, 10, 11, 12, 13\}$ qubits, respectively.

C. Platforms

The proposed Quantum-StateHD is implemented in both classical and quantum-dependent environments. Specifically, our approach is implemented in a classical environment using the PyTorch framework, while the related baseline approach is done with the help of the TorchHD [30] library. In a quantum environment, our approach is implemented through IBM's Qiskit platform [31], leveraging its quantum computing resources and simulation capabilities.

In order to ensure the comprehensiveness and reliability of the experimental results, we conduct several tests and comparative analyses on the performance of all methods in a classical environment, focusing on evaluating the accuracy, stability, and computational efficiency of the methods. In addition, we compare the performance of our methods in classical and quantum environments and analyze the influence of different environments on the performance of our methods. The dual-environment test validates the adaptability and robustness of

TABLE I

CLASSIFICATION ACCURACY OF QUANTUM-STATEHD AND OTHER HDC ALGORITHMS ON 5 DIFFERENT DATASETS. THE RESULTS SHOW THE AVERAGE OF THE TEN EXPERIMENTS AND THE STANDARD DEVIATION IS IN PARENTHESES, THE HIGHEST ACCURACY IS **BOLDED**, AND THE RUNNER-UP IS UNDERLINED.

	CARDIO10			CARDIO3			ISOLET			MNIST			UCIHAR		
	128	512	2048	128	512	2048	128	512	2048	128	512	2048	128	512	2048
AdaptHD	66.04 (3.10)	78.09 (3.52)	79.66 (2.93)	81.30 (4.54)	87.09 (3.91)	90.22 (2.78)	40.65 (2.07)	69.36 (4.01)	86.68 (0.62)	49.73 (8.19)	75.28 (2.07)	83.06 (7.21)	65.32 (0.78)	79.05 (4.78)	86.70 (2.70)
CompHD	44.05 (2.74)	56.73 (0.22)	57.51 (0.69)	65.18 (1.72)	67.53 (2.93)	72.93 (1.06)	36.84 (0.43)	70.09 (0.79)	83.39 (0.69)	49.68 (0.67)	70.06 (1.23)	80.00 (0.21)	61.25 (2.58)	75.36 (1.26)	81.81 (1.39)
DistHD	68.15 (2.15)	70.81 (3.60)	72.54 (2.21)	69.01 (5.31)	59.62 (6.93)	67.06 (6.82)	89.10 (0.59)	92.47 (0.61)	91.40 (1.39)	85.02 (0.59)	87.40 (0.33)	88.59 (0.44)	**92.83 (0.36)**	89.93 (4.89)	92.22 (2.59)
LeHDC	35.92 (7.47)	54.77 (6.46)	60.09 (4.89)	78.25 (4.50)	85.76 (0.44)	87.25 (1.25)	18.02 (2.28)	30.38 (4.24)	46.87 (1.66)	28.34 (6.07)	51.75 (3.66)	69.73 (2.04)	41.51 (2.19)	44.41 (2.89)	64.23 (1.04)
NeuralHD	69.56 (1.73)	77.93 (1.45)	78.33 (0.11)	84.59 (1.86)	88.18 (0.86)	87.87 (0.48)	87.21 (0.34)	91.36 (0.56)	92.28 (0.40)	82.07 (0.64)	87.35 (0.12)	88.21 (0.04)	87.26 (0.28)	89.20 (0.34)	89.80 (0.12)
OnlineHD	71.05 (1.23)	**78.33 (1.44)**	**81.22 (1.33)**	83.02 (2.21)	88.58 (1.28)	90.06 (2.31)	88.28 (0.70)	**92.50 (0.52)**	93.93 (0.26)	81.25 (0.52)	92.47 (0.41)	**95.51 (0.99)**	88.42 (1.94)	92.09 (0.98)	92.87 (0.39)
QuantHD	43.27 (0.77)	55.48 (3.81)	56.42 (0.44)	62.44 (3.27)	68.70 (2.40)	69.33 (0.80)	33.93 (4.11)	59.53 (0.71)	79.13 (0.59)	41.35 (3.32)	63.04 (0.72)	76.72 (0.79)	55.51 (1.60)	69.01 (1.86)	78.74 (1.56)
SparseHD	42.96 (5.85)	63.30 (3.62)	73.32 (1.09)	78.09 (3.28)	85.76 (0.62)	84.82 (2.46)	26.38 (1.31)	54.27 (1.19)	79.45 (0.76)	30.29 (5.92)	56.77 (3.42)	78.03 (1.23)	55.53 (2.56)	72.20 (3.53)	85.65 (1.03)
Vanilla	50.31 (2.55)	57.51 (3.37)	58.53 (0.95)	69.01 (4.49)	71.36 (0.84)	72.22 (1.46)	56.02 (1.39)	78.19 (0.23)	86.40 (0.43)	58.77 (3.06)	75.95 (0.43)	81.36 (0.52)	65.42 (0.92)	79.18 (0.37)	83.11 (0.87)
	7	9	11	7	9	11	7	9	11	7	9	11	7	9	11
Ours	**72.30 (2.50)**	76.62 (1.59)	80.28 (0.48)	**86.15 (1.39)**	**90.61 (0.98)**	**90.28 (0.65)**	**89.36 (1.74)**	92.15 (0.49)	**94.17 (0.37)**	**85.69 (0.47)**	**92.95 (0.14)**	93.08 (0.15)	88.12 (0.26)	**92.58 (1.25)**	**93.48 (0.87)**

TABLE II

QUANTITATIVE INFORMATION OF THE DATASETS

Dataset	# features	# classes	# training size	# test size	Description
CARDIO3	21	3	1700	426	Cardiac Signals Classification
CARDIO10	21	10	1700	426	Cardiac Signals Classification
MNIST	784	10	60000	10000	Handwritten Digit Recognition
ISOLET	617	26	6238	1559	Speech Recognition
UCIHAR	561	6	7352	2947	Human Activity Recognition

our approach while highlighting the potential advantages and limitations of quantum computing for specific tasks.

VI. RESULTS

A. Convergence

Fig. 4 illustrates the relationship between training accuracy and epochs, comparing the performance of four methods: LeHDC, OnlineHD, Ours (Original) implemented in a traditional Python environment, and Ours (Qiskit) implemented. Among these, Ours (original) achieves the best performance, with training accuracy rapidly increasing to nearly 100% within the first 5 epochs and maintaining this level throughout. Ours (Qiskit) also performs exceptionally well, showing a steady improvement in accuracy and finally surpassing OnlineHD, reaching greater than 95%. While OnlineHD achieves

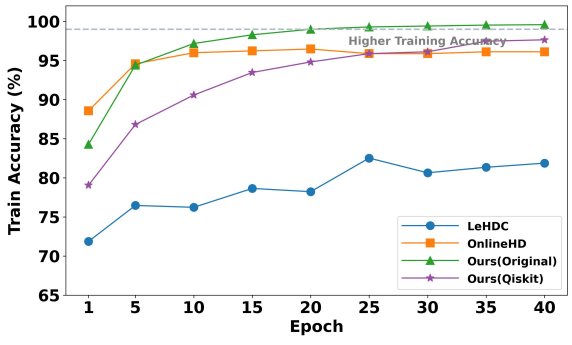

Fig. 4. Comparison of training processes between different models and implementations (case on CARDIO3 with dimension 128 or qubit 7)

high accuracy early on, it slightly underperforms compared to Ours (Qiskit) in the later stages. In contrast, LeHDC shows the weakest performance, with the accuracy gradually increasing and only reaching around 80% at the end. Overall, both Ours (Original) and Ours (Qiskit) demonstrate outstanding performance, with Ours (Original) achieving the highest accuracy and Ours (Qiskit) ultimately outperforming OnlineHD with its stable and reliable improvement. In general, our method demonstrates remarkable stability and rapid convergence during multiple training iterations, consistently outperforming other approaches in terms of both accuracy and training efficiency.

B. Accuracy

Table I shows the classification performance of various algorithms in five datasets (CARDIO10, CARDIO3, ISOLET, MNIST, UCIHAR) and different feature dimensions (128, 512, 2048). In general, as the dimension increases from 128 to 2048, the classification accuracy of all methods generally improves significantly, indicating that high-dimensional feature representation plays an important role in improving the classification effect. In comparison of different methods, the method proposed by our model performs best in the combination of various datasets and dimensions, showing a strong classification ability. For example, on the MNIST dataset, our method achieves 95.51% accuracy at 2048 dimensions, which is significantly higher than other methods. In the UCIHAR dataset, our method achieves an accuracy of 93.48% at 2048 dimensions, also much ahead of other methods. In the ISOLET dataset, our method also leads with an accuracy rate of 92.15%, but OnlineHD performs well on this data set, reaching 92. 47%. In contrast, traditional methods such as AdaptHD and DistHD perform well on some datasets (such as ISOLET and CARDIO10), but are still slightly inferior overall, while CompHD and QuantHD have mediocre classification performance and significantly lower accuracy on multiple datasets. Overall, our method shows strong generalization and stability, especially in the case of high-dimensional features, reflecting a breakthrough in the processing power of complex datasets.

In addition to implementing our algorithm with PyTorch, given that our algorithm involves quantum states and quantum coding, we also deployed it on IBM's Qiskit platform. The main difference between the two is the way the quantum states are implemented and fidelity calculations. Specifically, the implementation in the Qiskit platform stores and calculates quantum states through the state form in Qiskit. There are some differences in the results obtained by the two implementation methods, as shown in Table III.

The table presents the performance on the CARDIO3 dataset, with the seven dimensions indicating the number of quantum bits used in Qiskit, as shown in parentheses. Although both methods show good performance across different dimensions, the overall results are quite comparable. Therefore, our method demonstrates strong performance in both environments.

TABLE III
ACCURACY COMPARISON BETWEEN DIFFERENT PLATFORMS (CASE ON CARDIO3)

	128(7)	256(8)	512(9)	1024(10)	2048(11)	4096(12)	8192(13)
Qiskit	86.38	88.26	91.07	90.37	89.90	90.84	90.37
Classical	86.15	87.09	90.61	90.31	90.28	90.28	90.84

C. Time Consumption

TABLE IV
TIME CONSUMPTION (IN SECONDS) PER EPOCH OF DIFFERENT MODELS AT VARIOUS DIMENSIONS. (CASE ON CARDIO3 OURS-C MEANS CALCULATE THE SIMILARITY TRADITIONALLY)

	512(9)	1024(10)	2048(11)	4096(12)	8192(13)
Vanilla	0.0520	0.0626	0.102	0.2988	0.543
OnlineHD	0.0595	0.0832	0.0965	0.1001	0.120
LeHDC	0.0951	0.108	0.194	0.349	0.727
CompHD	0.0522	0.0606	0.107	0.316	0.626
Ours-C	0.130	0.195	0.221	0.256	0.457

To compare the complexity of our proposed method with existing approaches, we intuitively observe time consumption as a key metric. Table IV presents the time consumption per epoch (in seconds) for different models in various dimensions, using the CARDIO3 dataset as a case study. Our proposed method shows a higher time consumption compared to some existing approaches. However, it is important to emphasize that the increase in time consumption is not exponential. This validates that the computational complexity of our method on classical platforms remains comparable to current approaches and meets the low-complexity requirements of HDC. Furthermore, it is worth noting that if our method is implemented on actual quantum hardware or platforms, the time consumption is expected to decrease exponentially. This is due to the inherent advantages of quantum computing, such as parallelism and superposition, further demonstrating the potential efficiency improvements that can be achieved in future quantum implementations.

VII. CONCLUSION

In this paper, a hyperdimensional computation algorithm based on quantum state encoding is proposed. With this algorithm, hyperdimensional information of features can be extracted more efficiently. We both verify it theoretically and implement it in two ways: one in a classic Python environment, and the other experimentally in a Qiskit environment. The experimental results of the two implementation methods on different datasets are comparable to or even better than current mainstream state-of-the-art hyperdimensional computing algorithms. However, the experimental effect of the algorithm relying on Qiskit environment is slightly different from that of the classical method, which may be due to the difference in

simulation accuracy and the randomness of quantum simulation itself, so it is necessary to obtain a more accurate average value through multiple statistics. In addition, the storage and computing resources required by the method in quantum space are much lower than those of traditional hyperdimensional computing, and the reduction advantages are exponential. However, in both original Python and Qiskit environments, our experiments are based on simulations. In the future, we will aim to implement and test this approach in a real quantum environment.

REFERENCES

[1] G. Karunaratne, M. L. Gallo, M. Hersche, G. Cherubini, L. Benini, A. Sebastian, and A. Rahimi, "Energy efficient in-memory hyperdimensional encoding for spatio-temporal signal processing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1725–1729, 2021.

[2] P. Neubert, S. Schubert, and P. Protzel, "An introduction to hyperdimensional computing for robotics," *KI - Künstliche Intelligenz*, pp. 319 – 330, 2019.

[3] R. Thapa, B. Lamichhane, D. Ma, and X. Jiao, "Spamhd: Memory-efficient text spam detection using brain-inspired hyperdimensional computing," *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 84–89, 2021.

[4] S. Zhang, Z. Wang, and X. Jiao, "Adversarial attack on hyperdimensional computing-based nlp applications," *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2023.

[5] A. Hernández-Cano, Y. Ni, Z. Zou, A. Zakeri, and M. Imani, "Hyperdimensional computing with holographic and adaptive encoder," *Frontiers in Artificial Intelligence*, vol. 7, p. 1371988, 2024.

[6] A. Thomas, B. Khaleghi, G. K. Jha, S. Dasgupta, N. Himayat, R. Iyer, N. Jain, and T. Rosing, "Streaming encoding algorithms for scalable hyperdimensional computing," *ArXiv*, vol. abs/2209.09868, 2022.

[7] B. Khaleghi, J. Kang, H. Xu, J. Morris, and T. Simunic, "Generic: highly efficient learning engine on edge using hyperdimensional computing," *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022.

[8] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, pp. 139–159, 2009.

[9] J. Wang, S. Huang, and M. Imani, "DistHD: A learner-aware dynamic encoding method for hyperdimensional classification," *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2023.

[10] M. Imani, S. Bosch, S. Datta, S. Ramakrishna, S. Salamat, J. M. Rabaey, and T. Rosing, "QuantHD: A quantization framework for hyperdimensional computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 2268–2278, 2019.

[11] S. Duan, Y. Liu, S. Ren, and X. Xu, "LeHDC: Learning-based hyperdimensional computing classifier," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1111–1116.

[12] M. Imani, J. Morris, S. Bosch, H. Shu, G. De Micheli, and T. Rosing, "Adapthd: Adaptive efficient training for brain-inspired hyperdimensional computing," in *Proceedings of 2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2019, pp. 1–4.

[13] A. Hernández-Cano, N. Matsumoto, E. Ping, and M. Imani, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in *Proceedings of 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 56–61.

[14] J. Morris, M. Imani, S. Bosch, A. Thomas, H. Shu, and T. Rosing, "CompHD: Efficient hyperdimensional computing using model compression," in *Proceedings of 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[15] M. Imani, S. Salamat, B. Khaleghi, M. Samragh, F. Koushanfar, and T. Rosing, "SparseHD: Algorithm-hardware co-optimization for efficient high-dimensional computing," in *Proceedings of 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019, pp. 190–198.

[16] Z. Zou, Y. Kim, F. Imani, H. Alimohamadi, R. Cammarota, and M. Imani, "Scalable edge-based hyperdimensional learning system with brain-like neural adaptation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[17] D. Ranga, A. Rana, S. Prajapat, P. Kumar, K. Kumar, and A. V. Vasilakos, "Quantum machine learning: Exploring the role of data encoding techniques, challenges, and future directions," *Mathematics*, p. 3318, 2024.

[18] M. Rath and H. Date, "Quantum data encoding: a comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy," *EPJ Quantum Technology*, p. 72, 2024.

[19] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Expanding data encoding patterns for quantum algorithms," in *Proceedings of 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, 2021, pp. 95–101.

[20] C. Bravo-Prieto, "Quantum autoencoders with enhanced data encoding," *Machine Learning: Science and Technology*, p. 035028, 2021.

[21] Y.-G. Yang, X. Jia, S.-J. Sun, and Q.-X. Pan, "Quantum cryptographic algorithm for color images using quantum fourier transform and double random-phase encoding," *Information Sciences*, pp. 445–457, 2014.

[22] H. Yetiş and M. Karaköse, "An improved and cost reduced quantum circuit generator approach for image encoding applications," *Quantum Information Processing*, p. 203, 2022.

[23] B. Bhabhatsatam and S. Smanchat, "Hybrid quantum encoding: Combining amplitude and basis encoding for enhanced data storage and processing in quantum computing," in *Proceedings of 2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2023, pp. 512–516.

[24] S. R. Majji, A. Chalumuri, and B. S. Manoj, "Quantum approach to image data encoding and compression," *IEEE Sensors Letters*, pp. 1–4, 2023.

[25] X.-B. Nguyen, H.-Q. Nguyen, H. Churchill, S. U. Khan, and K. Luu, "Quantum visual feature encoding revisited," *Quantum Machine Intelligence*, p. 61, 2024.

[26] D. Campos and J. Bernardes, "Cardiotocography," UCI Machine Learning Repository, 2000, DOI: https://doi.org/10.24432/C51S4N.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[28] R. Cole and M. Fanty, "ISOLET," UCI Machine Learning Repository, 1991, DOI: https://doi.org/10.24432/C51G69.

[29] E. Bulbul, A. Cetin, and I. A. Dogru, "Human activity recognition using smartphones," in *Proceedings of 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2018, pp. 1–6.

[30] M. Heddes, I. Nunes, P. Vergés, D. Kleyko, D. Abraham, T. Givargis, A. Nicolau, and A. Veidenbaum, "Torchhd: An open source python library to support research on hyperdimensional computing and vector symbolic architectures," *Journal of Machine Learning Research*, pp. 1–10, 2023.

[31] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with Qiskit," 2024.