QiRBM: Optimized Restricted Boltzmann Machines using Quantum-Inspired Computing

Shanmuka Sadhu [1] **Minsung Kim** [1]

Abstract

Human Activity Recognition (HAR) systems often rely on deep learning models that suffer from high training latency and computational costs, limiting their deployment in dynamic, real-world environments. To address these inefficiencies, we introduce the Quantum-inspired Restricted Boltzmann Machine (QiRBM), a framework that replaces traditional Contrastive Divergence with a QUBO-formulated Parallel Tempering solver to escape local minima more effectively. Our approach integrates vectorized caching to eliminate redundant computations during Metropolis-Hastings sweeps and incorporates an adaptive temperature scheduler to ensure optimal replica exchange probabilities. Extensive evaluations on HAR time-series datasets demonstrate that QiRBM achieves a 92.9% reduction in training latency compared to classical RBM baselines while surpassing traditional LSTM models in classification accuracy. These results validate the potential of quantum-inspired optimization to significantly accelerate deep generative modeling for time-sensitive applications

1. Introduction

Human Activity Recognition(HAR) is a field of behavior analysis that's focused on interpreting human body motion using different methods such as sensors (Lara & Labrador, 2013) and network signals (Yousefi et al., 2017)to determine the physical actions of a person (Bulling et al., 2014). Applications of HAR include assisted living (Guerra et al., 2023), health monitoring (Li et al., 2023), smart homes (Bouchabou et al., 2021), security, and human–computer interaction (**?**). HAR systems are based on machine learning models trained on temporal sensor data, such as WiFi Channel State Information(CSI) (Wang et al., 2023; Salehinejad & Valaee, 2022),accelerometer or wearable IMU signals (Lara & Labrador, 2013), or vision-based modalities (Shin et al., 2024). Some HAR methods can be dynamically trained by actively adding newly observed data into the training set (Stikic et al., 2009). Current time-series methods for HAR include LSTM(Yousefi et al., 2017), CNNs(Yang et al., 2015), and U-Net(Zhang et al., 2018). These neural networks require large datasets, extensive hyperparameter tuning, and can be computationally expensive, causing high training latency. Current methods are not suitable for HAR in real-world settings since their high training latency can be difficult for dynamically changing environments.

Restricted Boltzmann Machines (RBMs) are probabilistic, energy-based models designed to capture complex data distributions through latent binary variables (Smolensky, 1986). RBMs can learn hidden units that capture underlying patterns of HAR data, which can be high-dimensional and noisy. RBMs have historically served as components in deep belief networks and early deep learning pipelines (Hinton et al., 2006; Hinton, 2012). RBMs can be used for HAR since they can be trained quick enough to adapt to dyanmically changing environments.

The energy landscape of the RBM is typically rugged, containing many isolated modes separated by large energy barriers (Salakhutdinov & Hinton, 2009). RBMs are dependent on sampling methods to find parameters that maximize the likelihood of the input. Current classical sampling methods, such as Gibbs sampling and Contrastive Divergence (CD), often become trapped in local minima, producing biased gradient estimates and limiting the RBM's ability to learn meaningful representations (Hinton, 2002). Although Gibbs sampling and Contrastive Divergence are designed to already be efficient, they are still computationally expensive methods. These issues become especially severe as models scale in size, depth, or complexity.

To prevent these issues, quantum annealing(QA) has become a popular approach for efficiently finding minima in the energy landscape (Kadowaki & Nishimori, 1998). Quantum annealers draw representative samples from a Boltzmann

[1]Department of Computer Science, Rutgers University, New Brunswick, New Jersey, USA. Correspondence to: Shanmuka Sadhu <shanmuka.sadhu@rutgers.edu>, Minsung Kim <minsungk.cs@rutgers.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

distribution by utilizing quantum tunneling to facilitate transitions through (rather than over) high barriers in the energy landscape. QA has computational limitations, particularly as problem sizes increase, which can restrict the number of usable variables (Johnson et al., 2011). Additionally, QA requires specialized hardware, which can be difficult to get and use (Johnson et al., 2011).

Classical parallel tempering (Geyer, 1991; Earl & Deem, 2005) offers a more scalable alternative, but it still suffers from suboptimal mixing when temperature schedules are not well-tuned. Our key insight is that quantum-inspired sampling can be take advantage of the many benefits of quantum annealing while remaining efficient on classical hardware. By reformulating RBM sampling as a parallel tempering process, it is possible to achieve faster exploration of the energy landscape and significantly lower sampling latency compared to traditional Gibbs-based methods.

In this work, we introduce **QiRBM**(**Q**uantum-**i**nspired **R**estricted **B**oltzmann **M**achine), a quantum-inspired framework for sampling during RBM training that improves efficiency and enables more effective learning in high-dimensional energy landscapes. Our key contributions are as follows:

- **Quantum-Inspired Parallel Tempering Solver:** We replace the traditional Contrastive Divergence (CD) algorithm with a QUBO-formulated Parallel Tempering solver. The QiRBM is able to escape local minima in the energy landscape more effectively, in terms of latency and accuracy, than classical Gibbs sampling.

- **Vectorized Caching:** We introduce a caching mechanism for vectorizing the spin-flip and replica-swap processes that eliminates redundant computations during Metropolis-Hastings sweeps.

- **Adaptive Temperature Control:** We incorporate an adaptive temperature scheduler that dynamically adjusts inverse temperatures to maintain constant energy overlap between replicas, ensuring uniform swap acceptance probabilities and faster convergence.

We evaluated the QiRBM and our baselines on an HAR time-series dataset(Yousefi et al., 2017) on training time and accuracy. The fully optimized QiRBM outperforms all our baselines in both training latency and evaluation accuracy, with a 92.9% reduction in latency compared to a classical RBM baseline (2.88s vs. 41.14s) and surpasses the classical LSTM baselines in accuracy (90.6% vs. 91.67%).

2. Background

2.1. Human Activity Recognition

Human Activity Recognition (HAR) are systems that identify human movements, such as walking, sitting, standing, running, or falling, by analyzing environmental sensor data. Applications of HAR include health monitoring for elderly individuals (Bhattacharya et al., 2022), surveillance applications (Taha et al., 2015), and fall detection (Li et al., 2019). Traditional HAR methods utilize wearable devices equipped with sensors that measure movement, such as accelerometers and gyroscopes (Bulling et al., 2014). Although these systems achieve high accuracies, they require users to carry a device, which can be impractical for real-world settings. Additionally, Camera-based HAR has emerged as an alternative; however, it requires a line of sight and may raise privacy concerns in many environments (Poppe, 2010). WiFi wireless signals have been used as a device-free approach to capture human activity in indoor environments(Abdelnasser et al., 2015). This works since human movements are able to disturb the wireless signals, causing changes in the channel state information(CSI).

2.2. Restricted Boltzmann Machines

A Boltzmann Machine (Ackley et al., 1985) is a stochastic generative model that models the probability distribution of the data. It is an undirected network with arbitrary pairwise connections between both its visible and hidden layer. The Boltzmann Machine learns to adjust its weights and biases to match the probability distribution of the input data. As demonstrated in Figure 1, A Restricted Boltzmann Machine(RBM) (Smolensky, 1986) is a variant of a Boltzmann machine where there exist only connections between hidden and visible layers, thus creating a bipartite structure. In the classical RBM formulation, both visible and hidden units are binary-valued random variables $v_i, h_j \in \{0, 1\}$.

Motivation. The RBM learns the likelihood different training inputs are by assigning low energy to parameters that resemble the training data and high energy to those that do not. Our input data corresponds to the visible nodes $\mathbf{v}$ of the RBM, as it's the observed data, and the learned representation of the RBM corresponds to the hidden nodes $\mathbf{h}$.

Since RBMs are energy-based probabilistic models (Hinton, 2012), each state is associated with the following energy calculation (Hopfield, 1982):

$$E(\mathbf{v}, \mathbf{h}, \mathbf{W}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij} \quad (1)$$

where $\mathbf{v}$ and $\mathbf{h}$ denote binary vectors for the visible and hid-

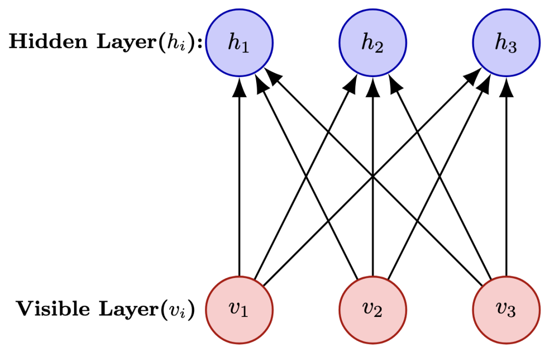

Figure 1. Diagram of Restricted Boltzmann Machines: There RBM consists of 2 layers: visible(red), and hidden(blue). Every visible unit is connected to every hidden unit, which creates a bipartite structure.

den layers, n_h and n_v are the number of units in **h** and **v** respectively, **a** and **b** are their corresponding bias vectors, and $\mathbf{W} = [w_{ij}] \in \mathbb{R}^{n_v \times n_h}$ is the weight matrix representing the interaction strength between visible unit v_i and hidden unit h_j. The energy decreases when visible and hidden units are aligned with the weights, meaning the model prefers those configurations.

Computing Visible and Hidden units. In order to calculate the joint probability $p(\mathbf{v}, \mathbf{h})$, we would need to sum over all combinations of **v** and **h**, which would require summing over $2^{n_v+n_h}$ states, which is computationally expensive.

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}, \quad Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{2}$$

Although RBMs bipartite structure allows us to calculate tractable conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$, the marginal probability $p(\mathbf{v})$ for exact maximum likelihood estimation is intractable due to Z. Instead of trying to compute the joint probability $p(\mathbf{v}, \mathbf{h})$, it is assumed that within a layer, the units are conditionally independent of each other (Hinton, 2012) since there exists no connections within a layer, only connections with nodes in the other layer. The probability of activation of each hidden unit i and visible unit j can be calculated using the logistic sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i w_{ij} v_i\right) \tag{3}$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j w_{ij} h_j\right) \tag{4}$$

Both Eq. 3 and Eq. 4 are both used during training every time RBM needs to compute hidden or visible units.

Loss Calculation. Although RBMs define a joint probability distribution $p(\mathbf{v}, \mathbf{h})$, in order for the RBM to learn the training data, the parameters $\theta = \mathbf{a}, \mathbf{b}, \mathbf{W}$ need to be adjusted. The training objective of an RBM requires finding parameter values that maximize the likelihood of the observed data and minimizes the energy function. This converts the probabilistic formulation of the RBM into an optimization problem. To determine the optimal values for $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{W}$, we can create a loss function. We can maximize the marginal log-likelihood of the input data $\mathbf{v}_{data}$ (Carreira-Perpinan & Hinton, 2005):

$$L_{gen} = \log p(\mathbf{v_{data}}) = \log \sum_{\mathbf{h}} e^{-E(\mathbf{v_{data}},\mathbf{h})} - \log Z \tag{5}$$

Since we never observe h,

$$p(\mathbf{v}_{data}) = \sum_{\mathbf{h}} p(\mathbf{v}_{data}, \mathbf{h}) \tag{6}$$

We can optimize the loss function L_{gen} using stochastic gradient descent(SGD). The gradient of L_{gen} decomposes into a positive phase and negative phase (Hinton, 2002):

$$\frac{\partial\big(-\log p(\mathbf{v}_{data})\big)}{\partial\theta} = \underbrace{\mathbb{E}_{\mathbf{h}\sim p(\mathbf{h}|\mathbf{v}_{data})}\left[\frac{\partial E(\mathbf{v}_{data}, \mathbf{h})}{\partial\theta}\right]}_{\text{positive phase}} - \underbrace{\mathbb{E}_{\mathbf{v},\mathbf{h}\sim p(\mathbf{v}_{data},\mathbf{h})}\left[\frac{\partial E(\mathbf{v}_{data}, \mathbf{h})}{\partial\theta}\right]}_{\text{negative phase}} \tag{7}$$

where $\theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}\}$ denotes the set of RBM parameters.

The positive phase strengthens the connection between the visible layer and the hidden units by lowering the network's energy for the input data. The negative phase weakens the connection for the RBM-generated data by raising the energy of the visible and hidden units.

Contrastive Divergence and Gibbs Sampling. The positive phase of the gradient requires computing expectations over the conditional distribution of the hidden units given the visible units(**v**), $p(\mathbf{h} \mid \mathbf{v})$. This isn't very difficult because the hidden units of the RBM are conditionally independent given the visible vector $\mathbf{v}$. Thus, the expectation can be computed as follows:

$$\mathbb{E}_{\mathbf{h}\sim p(\mathbf{h}|\mathbf{v})}[\mathbf{v}\mathbf{h}^T] = \mathbf{v} \cdot p(\mathbf{h}^T \mid \mathbf{v}).$$

On the other hand, the negative phase is calculated as the expectation over the joint distribution $p(\mathbf{v}, \mathbf{h})$:

$$\mathbb{E}_{\mathbf{v},\mathbf{h}\sim p(\mathbf{v},\mathbf{h})}[\mathbf{v}\mathbf{h}^T].$$

This is more difficult since $p(\mathbf{v}, \mathbf{h})$ depends on all RBM possibilities, which requires summing over all possible $\mathbf{v}$ and $\mathbf{h}$. Gibbs sampling (Hinton, 2002) can iteratively sample v and h for k iterations such that:

$$\mathbf{h}^{(k)} \sim p(\mathbf{h} \mid \mathbf{v}^{(k)}), \quad \mathbf{v}^{(k+1)} \sim p(\mathbf{v} \mid \mathbf{h}^{(k)}),$$

The samples $(\mathbf{v}^{(k)}, \mathbf{h}^{(k)})$ can approximate the distribution $p(\mathbf{v}, \mathbf{h})$, which provides an estimate for the negative phase:

$$\mathbb{E}_{\mathbf{v},\mathbf{h}\sim p(\mathbf{v},\mathbf{h})}[\mathbf{v}\mathbf{h}^T] \approx \mathbf{v}^{(k)} \cdot h^{(k)T}.$$

The k-step approximation calculation of the negative phase is called Contrastive Divergence(Hinton, 2002).

Updating RBM Parameters. We can finally update the weight and bias parameters using the given gradient.

$$\mathbf{W} \leftarrow \mathbf{W} + \alpha \left(\mathbf{v}_{\text{data}}\mathbf{h}_{\text{data}}^{\top} - \mathbf{v}^{(k)}\mathbf{h}^{(k)T}\right) \tag{8}$$

$$\mathbf{a} \leftarrow \mathbf{a} + \alpha \left(\mathbf{v}_{\text{data}} - \mathbf{v}^{(k)}\right) \tag{9}$$

$$\mathbf{b} \leftarrow \mathbf{b} + \alpha \left(\mathbf{h}_{\text{data}} - \mathbf{h}^{(k)}\right) \tag{10}$$

where α is the learning rate, $\mathbf{v}_{data}$ is in the input training data, and we sample $h_{\text{data}} \sim p(\mathbf{h} \mid \mathbf{v}_{\text{data}})$. By minimizing the loss function and updating the parameters of the RBM function, the RBM learns to assign higher probability (lower energy) to the training data.

3. Methods

3.1. Performing Classification with RBMs

Since a Restricted Boltzmann Machine is an unsupervised generative algorithm, it is not possible for a standalone RBM to perform a classification task. Thus we decided to use a Hybrid Discriminative RBM(HDRBM) inspired by (Larochelle & Bengio, 2008) to integrate a classification loss. The RBM first learns an unsupervised representation of the data by maximizing its log-likelihood(Eq. 5) to obtain the generative loss term L_{gen}. Although (Larochelle & Bengio, 2008) use a logistic sigmoid to find calculate their L_{disc}, we apply a linear transformation on top of a sigmoid to extract the hidden activations of the RBM:

$$\mathbf{h} = \sigma(\mathbf{W}^{\top}\mathbf{v} + \mathbf{b}), \tag{11}$$

where $\mathbf{v}$ is the input training data vector, $\mathbf{W}$ and $\mathbf{b}$ are the RBM weights and hidden layer biases, and $\sigma(\cdot)$ is the logistic sigmoid function. We then place these hidden activations into a multi-layer classifier to obtain logits $\mathbf{l}_t$:

$$\mathbf{l}_t = f_{\text{clf}}(\mathbf{h}), \tag{12}$$

where f_{clf} represents the classifier network (a linear layer, ReLU to introduce nonlinearity, dropout to prevent overfitting, and a final linear layer). We use a multi-layer classifier since, on its own, a raw sigmoid cannot easily capture complex relationships between features and multiple classes. To obtain the discriminative loss, we use a one-hot encoded representation of the true labels $\mathbf{y}$ and cross-entropy loss (Goodfellow et al., 2016) between the logits($\mathbf{l}_t$) and the labels($\mathbf{y}$) to get L_{disc}:

$$L_{\text{disc}} = -\sum_{c=1}^{C} y_c \log \hat{y}_c, \quad \hat{\mathbf{y}} = \text{softmax}(\mathbf{l}_t), \tag{13}$$

where C is the number of classes required for classification.

Our final loss is calculated as:

$$L_{total} = \alpha L_{gen} + L_{disc} \tag{14}$$

where α is a ratio applied on L_{gen}.

3.2. QUBO Formulation for RBM

Quadratic Unconstrained Binary Optimization (QUBO) is a mathematical representation of combinatorial optimization problems for physics-inspired computing (Kochenberger et al., 2014; Lucas, 2014). For a QUBO problem, the goal is to calculate a binary vector $x \in \{0, 1\}^n$ that minimizes a quadratic objective function of the form $x^T Q x$, where Q is a symmetric matrix encoding the optimization problem.

Classical Gibbs sampling can be a bottleneck for large RBMs; thus, a QUBO solver can be employed to sample more efficiently from the RBM distribution. (Adachi & Henderson, 2015) Since RBM is trained using an energy function, we can map the energy function to a QUBO matrix to use for solving via parallel tempering. The QUBO matrix Q is constructed by augmenting the RBM's weight and bias parameters as:

$$Q = \begin{bmatrix} \text{diag}(a) & W \\ W^{\top} & \text{diag}(b) \end{bmatrix} \tag{15}$$

where the diagonal blocks encode biases for visible and hidden units, and the off-diagonal blocks encode pairwise interactions. Then the optimization problem to be solved is

$$\min_{x\in\{0,1\}^{n_v+n_h}} x^{\top}(-Q)x \tag{16}$$

where the first n_v entries of x correspond to visible units and the remaining n_h entries correspond to hidden units (Kochenberger et al., 2014). Now, without relying on classical Gibbs sampling, this formulation allows sampling from the RBM distribution efficiently by using classical heuristic solvers like parallel tempering and quantum annealing (Adachi & Henderson, 2015).

3.3. Quantum-Inspired Parallel Tempering Solver for RBM Training

Rather than performing RBM training using Contrastive Divergence, as mentioned in previous works (Hinton, 2002; 2012), we employ quantum-inspired parallel tempering to efficiently obtain higher quality samples from the RBM distribution.

We construct a QUBO formulation of the RBM from Equation. 15 where off-diagonal blocks contain W and W^T and diagonal entries contain visible and hidden biases.

We work in the QUBO domain with Boolean variables such that each replica r corresponds to a spin state $\mathbf{s}_r \in \{0, 1\}$ for both hidden and visible variables. Each replica is associated with an inverse temperature $\beta_r = 1/T_r$. The β_r vector is defined by the minimum temperature $T_{\min}$ and maximum temperature $T_{\max}$:

$$\beta_r = \frac{1}{T_{\min}\,\rho^{r-1}}, \qquad \rho = \exp\left(\frac{\ln(T_{\max}/T_{\min})}{R-1}\right). \quad (17)$$

where there are R total parallel replicas. For each replica, a Metropolis-Hastings sweep (Metropolis et al., 1953) is performed to update the individual spin state. The QUBO matrix is decomposed into local fields (diagonal terms) and couplings (off-diagonal terms), consistent with spin-based QUBO formulations. We define the change in energy(ΔE) as:

$$\Delta E_k = (1 - 2s_k)h_k \quad (18)$$

which corresponds to flipping a bit $s_k \to 1 - s_k$

$$h_k = \sum_j Q_{kj} s_j. \quad (19)$$

and $\mathbf{Q}$ is the coupling matrix. Addtionally, the energy of the system is calculated as

$$E = \hat{E} + \Delta E \quad (20)$$

where $\hat{E}$ is the current energy of the system. We apply a spin-flip(i.e., from 0 to 1 or 1 to 0) if the change in energy $\Delta E < 0$. Otherwise if $\Delta E > 0$ we use an acceptance rule to determine the probability of flipping:

$$P_{\text{accept}} = \min\left(1, e^{-\beta_r \Delta E_k}\right). \quad (21)$$

At high temperatures (small β_r), the acceptance rule promotes more random exploration, while at lower temperatures (higher β_r), it promotes fewer uphill moves and focuses on lower energy states (Earl & Deem, 2005). Higher temperatures yield more spin flip acceptances, while lower temperatures yield less acceptances.

Parallel Tempering (PT), also known as replica-exchange Monte Carlo (Geyer, 1991), swaps neighboring replicas to promote global exploration of the energy landscape as simplified in Figure 2. We decide whether to swap neighbors if the metropolis criterion is accepted; thus, the swap rule is similar to the spin-flip proposal:

$$p_{\text{swap}} = \min\left(1, e^{(\beta_i - \beta_{i+1})(E_{i+1} - E_i)}\right) \quad (22)$$

Higher-temperature replicas encourage global exploration throughout the landscape while lower-temperature replicas encourage fine-tuned searches (Earl & Deem, 2005).

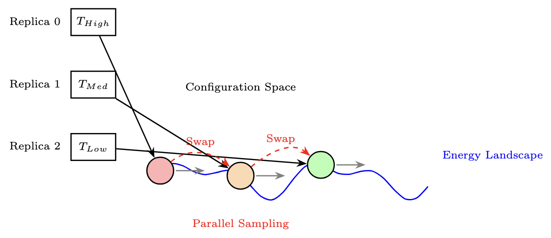

Figure 2. Illustration of Parallel Tempering: All replicas (colored circles) sample the same energy landscape at different temperatures. They are neighboring replicas and are only swapped if $\Delta E < 0$ or the acceptance criterion is true

After all sweeps and exchanges are completed, we select the RBM states that result in the lowest energy across all replicas as the negative phase sample $(v^{(k)}, h^{(k)})$ for updating RBM parameters via Equations(8-10). We select the replica with the lowest encountered energy as an approximation to the negative-phase expectation. Thus, we don't need to use the conventional Gibbs Sampling for training RBMs.

3.4. Vectorized Caching for RBM

In the approach mentioned in Section 3.3, for each sweep and every replica, we recalculate the local fields h, which is crucial for energy calculation. This results in a major bottleneck due to the redundant computation. Vectorized caching removes this bottleneck by computing all effective fields in a single batched matrix operation and reusing them throughout the sweep, eliminating repeated per-spin recalculation.

In vectorized caching, we initally compute $\mathbf{h}_{\text{cache}}$ jointly through a single matrix operation:

$$\mathbf{h}_{\text{cache}} = \mathbf{Q} \cdot \mathbf{s} \quad (23)$$

where $\mathbf{Q}$ is the couplings matrix(off-diagonal terms of QUBO matrix) and $\mathbf{s} \in \{0, 1\}$ is the current state spin vector. This formulation allows all local field updates to be cached in memory, being used later on during the sweep. Then the change in energy ΔE is computed in parallel for all spins and replicas as:

$$\Delta E = (1 - 2\,\mathbf{s}) \odot \mathbf{h}_{\text{cache}}, \quad (24)$$

where $\odot$ denotes element-wise multiplication. Since we perform h and ΔE in parallel, we also need to vectorize both the spin-flip updates and parallel tempering. We evaluate flip acceptance probabilities for all spins and all replicas simultaneously:

$$\mathbf{p}_{\text{flip}} = \exp(-\beta \otimes \Delta E) \tag{25}$$

where $\mathbf{p}_{\text{flip}}$ is the probability of a spin flip, $\boldsymbol{\beta}$ is the inverse temperature schedule, and $\otimes$ is the tensor product.

$$\mathbf{P}_{\text{flip}} = \mathbf{p}_{\text{flip}} > \mathcal{U}(0,1) \tag{26}$$

where $\mathcal{U}(0,1)$ denotes a uniform random tensor used across all replicas. The resulting boolean mask $\mathbf{P}_{\text{flip}}$ directly indicates which spins are to be flipped. Additionally, our states $\mathbf{s}$ are also updated using the boolean mask P_{flip}:

$$\mathbf{s} = \mathbf{s} \odot (1 - 2\,\mathbf{P}_{\text{flip}})\,, \tag{27}$$

Since we retain $\mathbf{h}_{\text{cache}}$, the local fields can be updated incrementally rather than recomputed. For each accepted flip at index i, we update

$$\mathbf{h}_{cache} \leftarrow \mathbf{h}_{cache} + 2\mathbf{Q}[:, i]s_i \tag{28}$$

When flipping a single unit s_i, the only weights that affect the energy change are those connected to that unit. In the QUBO representation, these weights correspond to the i-th column of $\mathbf{Q}$. This lets us compute energy changes without recomputing the entire energy function from scratch, using only the pre-cached weighted input $\mathbf{h}_{cache}$. With a vectorized ΔE, we can now simply update our energy vector as well:

$$\mathbf{E} \leftarrow \mathbf{E} + \Delta E \odot \mathbf{P}_{\text{flip}}. \tag{29}$$

3.4.1. Vectorized Parallel Tempering:

We can also perform vectorized parallel tempering similarly to how we performed vectorization for our spin-flip updates. We create the boolean mask $\mathbf{P}_{\text{flip}}$ as follows:

$$\mathbf{p}_{\text{swap}} = \exp[-(\beta_r - \beta_{r+1})(E_{r+1} - E_r)] \tag{30}$$

$$\mathbf{P}_{\text{swap}} = \mathbf{p}_{\text{swap}} > \mathcal{U}(0,1) \tag{31}$$

where $\mathbf{P}_{swap}$ is a boolean matrix mask, $\mathcal{U}(0,1)$ denotes a uniform random tensor applied to each replica r. All swap proposals and acceptances are evaluated in parallel via tensor operations, using preallocated buffers to minimize synchronization cost. Accepted replicas exchange both their spin configurations and energies in place.

The matrix is stored in a row-major layout, which improves cache locality during sweeps. By converting to complete vectorization while also using caching, we reduce the number of for-loops to one, as we only need to go through each sweep sequentially.

Algorithm 1 Metropolis Update with Vectorized Caching

```
1: for $t = 1$ to $T$ do
2:     Compute effective field: $\mathbf{h}_{\text{cache}} = Q \cdot s + b$
3:     Compute energy differences: $\Delta E = (2\,\mathbf{s} + 1) \odot \mathbf{h}_{\text{cache}}$
4:     $\mathbf{P_{flip}} \leftarrow \exp(-\boldsymbol{\beta} \otimes \Delta\mathbf{E}) > \mathcal{U}(0,1)$
5:     Flip accepted spins: $\mathbf{S}[\mathbf{F}] \leftarrow -\mathbf{S}[\mathbf{F}]$
6:     Update energies: $E \leftarrow E + \sum_i \Delta E \cdot \mathbf{P}_{\text{flip}}$
7:     for replica $r = 1$ to $R - 1$ do
8:         $P_{\text{swap}} = \exp[-(\beta_r - \beta_{r+1})(E_{r+1} - E_r)]$
9:         if $\mathcal{U}(0,1) < P_{\text{swap}}$ then
10:            Swap $\mathbf{S}_r \leftrightarrow \mathbf{S}_{r+1}$ and $E_r \leftrightarrow E_{r+1}$
11:        end if
12:    end for
13:    Update best state:
14:    if $\min(\mathbf{E}) < E^*$ then
15:        $E^* \leftarrow \min(\mathbf{E})$
16:        $\mathbf{S}^* \leftarrow \mathbf{S}[\arg\min(\mathbf{E})]$
17:    end if
18: end for
19: return $\mathbf{S}^*$
```

As shown in Algorithm 1, when combining caching with full vectorization, we reduce all per-spin operations to elementwise tensor updates, which leaves only the outer sweep loop as a sequential component.

3.5. Adaptive Replica Allocation for Parallel Tempering

To enhance sampling efficiency, we used an adaptive temperature scheduler based on the optimal replica allocation proposed by (Rathore et al., 2005). As described previously, traditional parallel tempering methods employ a fixed geometric or linear temperature spacing, which can result in nonuniform swap probabilities and inefficient exploration of the energy landscape (Geyer, 1991; Earl & Deem, 2005). Thus, we propose the use of Adaptive Replica Allocation for adaptively adjusting the inverse temperatures $\{\beta_i\}_{i=1}^{R}$ of replicas R such that the energy overlap between neighboring replicas remains approximately constant. By doing this, neighboring replicas will have nearly uniform swap acceptance probabilities (Rathore et al., 2005).

Let E_i and σ_i denote the mean and standard deviation of the energy distribution at inverse temperature β_i, respectively. We maintain a constant s normalized energy separation between adjacent replicas:

$$s = \frac{\Delta E_i}{\bar{\sigma}_i} = \frac{E_{i+1} - E_i}{(\sigma_{i+1} + \sigma_i)/2}, \tag{32}$$

which ensures sufficient overlap between $P(E;\beta_i)$ and $P(E;\beta_{i+1})$ for efficient exchanges. overlap sufficiently for efficient exchanges. A linear temperature scheduler is initially used to gather the initial E_i and σ_i values. After every n sweeps, we estimate $\{E_i, \sigma_i\}$ and update the intermediate inverse temperatures such that each replica pair satisfies the constant-overlap criterion in Eq. 32 We are able to minimize the variance of the exchange probability and maintain approximately uniform replica mobility across all temperatures (Rathore et al., 2005).

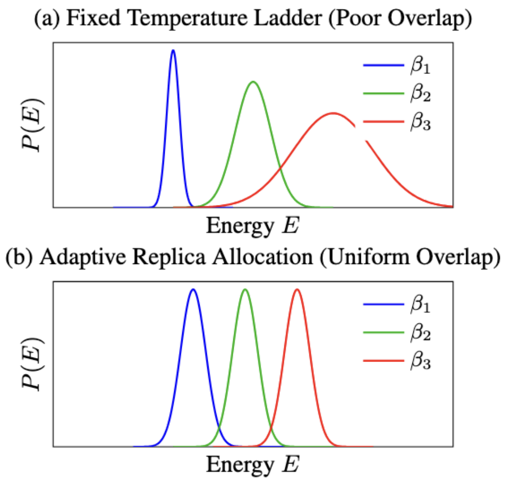

Figure 3. Energy distribution overlap for replicas at different temperatures. (a) Fixed geometric/linear spacing produces nonuniform overlap. (b) Adaptive replica allocation yields nearly constant overlap and improved swap probabilities.

This adaptive scheme maintains nearly uniform swap acceptance across replicas, reduces round-trip times, and accelerates convergence of the PT ensemble. This will allow for faster exploration of low-energy states and more consistent mixing compared to fixed ladder designs.

4. Experiments

In this section, we discuss our experiments and results compared to our baselines, and report an ablation study to determine the optimal configuration of our approach.

4.1. Experimental Setup

Dataset: We used the HAR data from (Yousefi et al., 2017). The data consists of six human tasks: lying down, falling, walking, running, sitting down, and standing up. The dataset consists of a total of 4973 samples, each containing 1,000 time steps recorded at 1 kHz, with 90 WiFi Channel State Information (CSI) features per time step. For our training set we use 3979 samples, which allows for 797 samples for our evaluation set and development set each.

Models: Our baselines include (1) the original LSTM architecture from (Yousefi et al., 2017), which we reproduced; (2) a classical discriminative RBM trained with Contrastive Divergence (Larochelle & Bengio, 2008); (3) the NASA PySA RBM (NASA, 2023) with a discriminative loss; and (iv). our QiRBM baseline which does not include improvements. Training is performed with a batch size of 512, target dimension 512, Adam optimizer with learning rate 1e-3, and 30 epochs.

Metrics: We tracked the **accuracy** on our evaluation set and the **training latency** of each model to determine the quickest model while maintaining the accuracy achieved without any of our proposed methods.

Hardware Setup: We use the Rutgers Amarel GPU cluster, requesting 1 L40S NVIDIA GPU and 1 Intel Xeon Gold 6538Y+ Processor

4.2. Pre-Processing:

The HAR data(Yousefi et al., 2017) originally has a shape of (4973, 1000, 90), but it is first downsampled to 500 Hz, resulting in a shape of (4973, 500, 90). Since our time series data consists of 4793 samples of shape (900,50), we need to input a 45000-element matrix. This can be computationally expensive and may be challenging for memory; therefore, we perform PCA to minimize information loss and reduce dimensionality. Our results comparing classical and quantum-inspired RBM consist of PCA to 1200 dimensions.

4.3. Ablation Study

For all experiments in the ablation study, we perform evaluation on the developmental set. Additionally, we use a minimum temperature of 1.0 and maximum temperature of 3.5. The goal of this ablation study is to isolate each hyperparameter(PCA dimension, sweeps, and replicas) and examine its impact on our proposed approach.

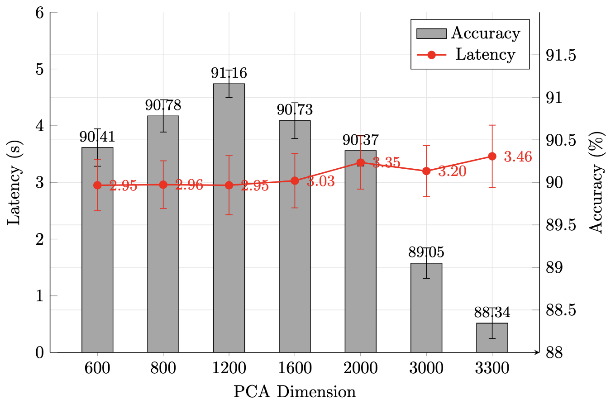

Figure 4. Ablation study for PCA Dimension.

Without PCA, we found that the latency was 28 seconds and the accuracy was 68%, which is significantly worse than our baselines. Since PCA can result in information loss, we wanted to test which PCA would yield minimal accuracy loss while maintaining quicker latency. Since the maximum number of PCA components is the minimum of the number of samples or the number of rows, we began with the maximum number of PCA components (3300) and proceeded to decrease the PCA until we found a suitable dimension.

For the PCA ablation, we chose 2 sweeps and 25 replicas. In Figure 4, the PCA dimension with the highest accuracy was 1200 at 91.16s. Although 600,800,1200 PCA all had very similar training latency measurements, 1200 PCA was chosen since it correlated with the highest accuracy as well.

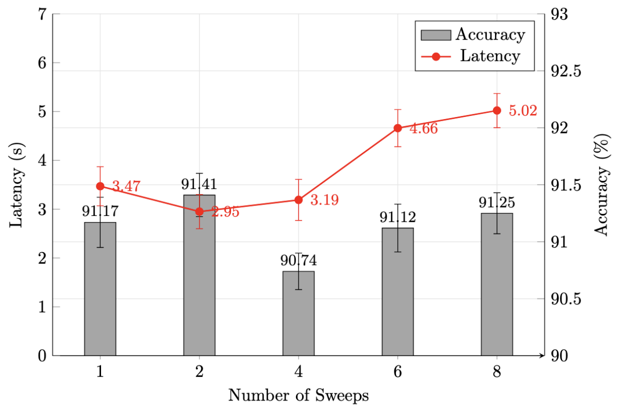

Figure 5. Ablation study for optimal sweeps.

For the sweep ablation study, we used 1200 PCA from Figure 4 and 25 replicas. In Figure 5, 2 sweeps shows the quickest latency of 2.95s and the highest accuracy of 91.41%. Thus, we chose to use 2 sweeps for the rest of the ablation and our final results. Our results showed that increasing the number of sweeps did not provide benefits for our problem. Figure 6 evaluates the effect of replica count using 1200 PCA components from Figure 4 and 2 sweeps from Table 5. Although 25 replicas provide the lowest latency (2.95 s), the highest accuracy (92.01%) occurs at 100 replicas. Since we prioritized latency, we chose 25 replicas for the other ablation study and final results.

In Table 1, we study the contribution each component to our final approach. The NASA PySA(NASA, 2023) implementation of RBM originally achieved a latency of 22.57 seconds and 89.05% accuracy. Each component proceeded to decrease latency and mostly improve accuracy. Adding vectorized caching alone reduces latency by 55.92% with no loss in accuracy. Vectorized PT also decreased latency by 30.32%. Since adaptive temperature control enables quicker convergence and optimal replication allocation, we

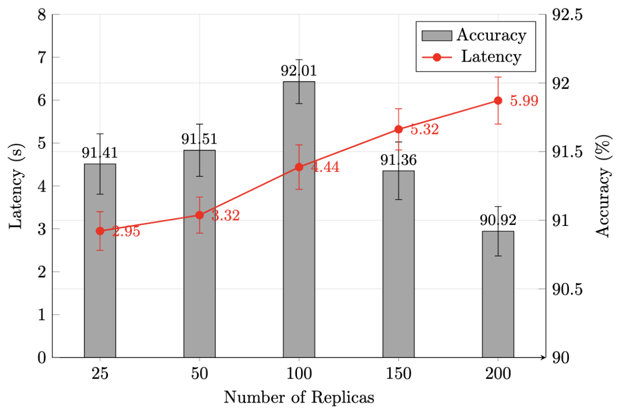

Figure 6. Ablation study for optimal replicas.

Table 1. Ablation of our Quantum-Inspired RBM

Approach	Latency(s)	Accuracy(%)
PySA RBM(NASA, 2023)	22.57	89.05
QiRBM-Baseline(Qi-B)	12.57	90.94
Qi-B + ΔC	5.54	90.65
Qi-B + ΔC + ΔPT	3.86	90.20
QiRBM:Qi-B + ΔC + ΔPT + Adaptive T	**2.95**	**91.41**

observed a 1-second increase in latency and a 1.2% increase in accuracy. For all experiments that did not use adaptive temperature control, a linear temperature scheduler was used instead.

4.4. Full Results

Table 2. Comparison of all approaches

Approach	Latency(s)	Accuracy(%)
LSTM (Yousefi et al., 2017)	3000	90.60
Classical RBM (Hinton, 2012)	41.14	88.35
PySA RBM (NASA, 2023)	22.42	90.01
Qi-B	12.12	90.01
QiRBM	**2.88**	**91.67**

From the ablation study, we used the best setup for our appraoch to compare to our baselines. In (Yousefi et al., 2017), the authors demonstrate that the best approach for their work is LSTM, achieving an accuracy of 90.5%. In Table 2, our reproduced LSTM also achieved 90.6% accuracy and a training time of 3000 seconds. The classical RBM recorded a substantial latency increase compared to the LSTM, at 41.14 seconds, but still performed worse in terms of accuracy. With both PySA RBM and QiRBM, the latency decreased to 22.42 seconds and 12.12 seconds, respectively. Finally, in Table 2, we demonstrate that our approach achieves 2.88 seconds and 91.67% accuracy, significantly outperforming all baselines.

5. Conclusion & Future Work

In this work, we introduced a fully optimized, quantum-inspired RBM training pipeline that significantly accelerates sampling while improving model accuracy. By integrating vectorized caching, vectorized parallel tempering, and adaptive temperature schedules, QiRBM achieves an 1328% reduction in latency compared to the classical PySA RBM baseline. Extensive ablation studies demonstrate that each proposed component contributes to both latency and performance, with the full system achieving an inference time of 2.95 seconds and an accuracy of up to 91.41%.

We hope to extend the QiRBM beyond Human Activity Recognition to different datasets. We believe this work will promote future research at the intersection of quantum-inspired algorithms and scalable deep generative modeling, particularly for applications that require fast and high-quality sampling.

References

Abdelnasser, H., Youssef, M., and Harras, K. A. Wigest: A ubiquitous wifi-based gesture recognition system. In *2015 IEEE conference on computer communications (INFOCOM)*, pp. 1472–1480. IEEE, 2015.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. A learning algorithm for boltzmann machines. *Cognitive Science*, 9 (1):147–169, 1985.

Adachi, S. H. and Henderson, M. P. Application of quantum annealing to training of deep neural networks. *arXiv preprint arXiv:1510.06356*, 2015.

Bhattacharya, D., Sharma, D., Kim, W., Ijaz, M. F., and Singh, P. K. Ensem-har: An ensemble deep learning model for smartphone sensor-based human activity recognition for measurement of elderly health monitoring. *Biosensors*, 12(6):393, 2022.

Bouchabou, D., Nguyen, S. M., Lohr, C., LeDuc, B., and Kanellos, I. A survey of human activity recognition in smart homes based on iot sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18):6037, 2021.

Bulling, A., Blanke, U., and Schiele, B. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 2014.

Carreira-Perpinan, M. A. and Hinton, G. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pp. 33–40. PMLR, 2005.

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Geyer, C. J. Markov chain monte carlo maximum likelihood. 1991.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Guerra, B. M. V., Torti, E., Marenzi, E., Schmid, M., Ramat, S., Leporati, F., and Danese, G. Ambient assisted living for frail people through human activity recognition: state-of-the-art, challenges and future directions. *Frontiers in neuroscience*, 17:1256682, 2023.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Hinton, G. E. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade: Second Edition*, pp. 599–619. Springer, 2012.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18 (7):1527–1554, 2006.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Johnson, M. W., Amin, M. H., Gildert, S., et al. Quantum annealing with manufactured spins. *Nature*, 473(7346): 194–198, 2011.

Kadowaki, T. and Nishimori, H. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H., and Wang, Y. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, 2014.

Lara, O. D. and Labrador, M. A. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013.

Larochelle, H. and Bengio, Y. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pp. 536–543, 2008.

Li, A., Bodanese, E., Poslad, S., Chen, P., Wang, J., Fan, Y., and Hou, T. A contactless health monitoring system for vital signs monitoring, human activity recognition, and tracking. *IEEE Internet of Things Journal*, 11(18): 29275–29286, 2023.

Li, H., Shrestha, A., Heidari, H., Le Kernec, J., and Fioranelli, F. Bi-lstm network for multimodal continuous human activity recognition and fall detection. *IEEE Sensors Journal*, 20(3):1191–1201, 2019.

Lucas, A. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

NASA. Pysa: Python solver for optimization of classical cost functions. https://github.com/nasa/PySA, 2023. Version as of 2023, United States Government work, NASA.

Poppe, R. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

Rathore, N., Chopra, M., and de Pablo, J. J. Optimal allocation of replicas in parallel tempering simulations. *The Journal of chemical physics*, 122(2):024111, 2005.

Salakhutdinov, R. and Hinton, G. Deep boltzmann machines. In *Artificial intelligence and statistics*, pp. 448–455. PMLR, 2009.

Salehinejad, H. and Valaee, S. Litehar: Lightweight human activity recognition from wifi signals with random convolution kernels. *arXiv preprint arXiv:2201.09310*, 2022.

Shin, J., Hassan, N., Miah, A. S. M., and Nishimura, S. A comprehensive methodological survey of human activity recognition across diverse data modalities. *Sensors*, 2024.

Smolensky, P. Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, pp. 194–281. MIT Press, Cambridge, MA, 1986.

Stikic, M., Huynh, T., Van Laerhoven, K., and Schiele, B. On-line activity recognition in ambient assisted living environments. In *International Conference on Ambient Assisted Living*, pp. 1–8. Springer, 2009.

Taha, A., Zayed, H. H., Khalifa, M. E., and El-Horbaty, E.-S. M. Human activity recognition for surveillance applications. In *Proceedings of the 7th International Conference on Information Technology*, pp. 577–586, 2015.

Wang, Y. et al. Csi-based location independent human activity recognition using deep learning. *Human-Centric Intelligent Systems*, 2023.

Yang, J., Nguyen, M. N., San, P. P., Li, X., Krishnaswamy, S., et al. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pp. 3995–4001. Buenos Aires, Argentina, 2015.

Yousefi, S., Narui, H., Dayal, S., Ermon, S., and Valaee, S. A survey on behavior recognition using wifi channel state information. *IEEE Communications Magazine*, 55(10): 98–104, 2017.

Zhang, Y., Zhang, Y., Zhang, Z., Bao, J., and Song, Y. Human activity recognition based on time series analysis using u-net. *arXiv preprint arXiv:1809.08113*, 2018.