Winning Lottery Tickets in Neural Networks via a Quantum-Inspired Classical Algorithm

Natsuto Isogai
The University of Tokyo
natsuto.isogai@g.ecc.u-tokyo.ac.jp

Hayata Yamasaki
The University of Tokyo
hayata.yamasaki@gmail.com

Sho Sonoda
RIKEN AIP and
CyberAgent

Mio Murao
The University of Tokyo

Abstract

Quantum machine learning (QML) aims to accelerate machine learning tasks by exploiting quantum computation. Previous work studied a QML algorithm for selecting sparse subnetworks from large shallow neural networks. Instead of directly solving an optimization problem over a large-scale network, this algorithm constructs a sparse subnetwork by sampling hidden nodes from an optimized probability distribution defined using the ridgelet transform. The quantum algorithm performs this sampling in time $O(D)$ in the data dimension D, whereas a naive classical implementation relies on handling exponentially many candidate nodes and hence takes $\exp[O(D)]$ time. In this work, we construct and analyze a quantum-inspired fully classical algorithm for the same sampling task. We show that our algorithm runs in time $O(\mathrm{poly}(D))$, thereby removing the exponential dependence on D from the previous classical approach. Numerical simulations show that the proposed sampler achieves empirical risk comparable to exact sampling from the optimized distribution and substantially lower than sampling from the non-optimized uniform distribution, while also exhibiting exponentially improved runtime scaling compared with the conventional classical implementation. These successful dequantization results show that sparse subnetwork selection via optimized sampling can be achieved classically with polynomial data-dimension scaling on conventional computers without quantum hardware, providing an alternative to the existing quantum algorithm.

1 Introduction

State-of-the-art neural networks achieve strong predictive performance, but their size makes training increasingly costly. This has led to extensive interest in identifying sparse trainable subnetworks that preserve much of the performance of the original dense networks. Representative routes include pruning after training [1], pruning at initialization [2, 3], and dynamic sparse training [4]. The lottery ticket hypothesis further suggests that large randomly initialized dense networks may contain sparse subnetworks that can be trained in isolation and achieve performance comparable to that of the original dense network [5, 6].

Many existing routes to sparse subnetworks are formulated as optimization procedures over network weights, and thus, sparse-subnetwork construction is often treated as a search problem over weights, masks, or connectivity patterns by handling the original neural network. By contrast, the random feature method [7–9], originally developed for accelerating kernel methods, motivates a different route for constructing such sparse networks. Ridgelet representations of shallow, single-hidden-layer

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

neural networks provide a natural analytic viewpoint for such a route [10]. In the ridgelet viewpoint, the target function is represented as a linear combination of basis functions corresponding to nodes in the hidden layer, and the corresponding coefficients quantify the importance of each hidden node. This makes sparse-subnetwork construction amenable to importance sampling. Rather than searching for a sparse subnetwork, one may define a probability distribution that places larger mass on more important hidden nodes and construct a sparse network directly by drawing samples from this distribution. Each sample corresponds to one hidden node, and thus, the constructed network size is determined by the number of samples.

A realization of this idea was previously proposed in [11] using quantum computation, where the authors introduce the quantum ridgelet transform and use it to construct sparse single-hidden-layer neural networks. Their approach defines an optimized probability distribution over hidden-node parameters, using given data. Then, it samples from this distribution to construct a sparse network. This provides a sampling-based route to sparse-subnetwork construction, motivated by the lottery ticket hypothesis but distinct from standard pruning-based routes.

The existing approach to this sampling procedure relies on quantum computation. Quantum machine learning (QML) has been studied as a field aiming to accelerate machine learning by using quantum computation, which has the capability of implementing high-dimensional linear transformations via the technique of quantum singular-value transformation (QSVT) [12–15]. In the ridgelet-based subnetwork construction of [11], quantum computation is used for two operations: implementing the ridgelet transform and applying QSVT to implement the inverse of a matrix with regularization, which is required to define the optimized sampling distribution. However, an implementation of such a QML algorithm requires quantum hardware capable of preparing and manipulating coherent quantum states. Large-scale fault-tolerant quantum computation (FTQC) capable of implementing such primitives is not yet achieved on current quantum hardware, and realistic implementations may incur substantial overhead from noise suppression and quantum error correction [16, 17]; therefore, a classical, i.e., non-quantum, algorithm running on conventional computational hardware for the same probability distribution may be valuable when its overhead is moderate, e.g., polynomial overhead.

This viewpoint has also motivated extensive work on quantum-inspired classical algorithms, which asks whether algorithmic ideas originally developed for quantum computation can be converted into efficient classical algorithms with only moderate overhead. In general, classical simulation of quantum computation is not always expected to be efficient, and this belief is one of the motivations for quantum computation [18, 19]. Nevertheless, a series of quantum-inspired classical algorithms has shown that, under suitable input models or structural assumptions, certain machine-learning tasks initially solved by polynomial-time quantum algorithms can also be solved by polynomial-time classical algorithms [20–22]. In this sense, we say that a quantum algorithm is "dequantized" when its computational task is solved by a classical algorithm with at most polynomial overhead.

These results suggest that when a quantum algorithm relies on the data-access assumption or task-specific structure, the corresponding sampling procedure or linear-algebraic transformation may sometimes admit an efficient classical algorithm. This motivates the central question of this work: whether quantum computation is essential for the above sampling, i.e., whether a classical algorithm can achieve the sampling task with a performance comparable to the quantum algorithm in [11].

1.1 Problem

The ridgelet-based route turns sparse-subnetwork construction into a sampling task. The optimized distribution of [11] assigns probability mass to hidden-node parameters according to the magnitudes of their regularized ridgelet coefficients. Sampling from this distribution is a natural way to select important hidden nodes, where each sampled parameter becomes one node in the sparse network.

The primary difficulty in this sampling task is that the regularized ridgelet coefficients are not directly available. They are expressed through an inverse of a matrix with regularization

$$(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^{\top} + \lambda P^{-D}\boldsymbol{I})^{-1}\boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle\,, \tag{1}$$

where $\boldsymbol{R}$ denotes the discrete ridgelet transform, and $\hat{\boldsymbol{P}}_{\mathrm{data}}$ is a matrix determined by the empirical data distribution. This inverse acts on the space of hidden-node parameters, whose size grows exponentially with the dimension D of the input data. Moreover, the matrix to be inverted is not sparse or low rank in general. Thus, a direct classical implementation involves computing the inverse of a dense $\exp[O(D)] \times \exp[O(D)]$ matrix, which usually takes exponential time in D.

The second obstacle is that the ability to compute the unnormalized weight at each hidden node is not enough to obtain a sampler. Even if one can compute the score of a candidate hidden node, sampling from the optimized distribution would still require normalizing these weights over the entire space. The normalization constant is a sum over all hidden-node parameters and thus cannot be efficiently computed by direct enumeration over the exponentially large parameter space. Furthermore, a naive rejection sampler based on a uniform proposal distribution cannot be efficient since the target distribution may be highly concentrated on a small subset of important nodes.

The quantum algorithm in [11] overcomes these obstacles by implementing the relevant high-dimensional transformations as operations on quantum states rather than manipulating classical vectors. Using the quantum ridgelet transform and QSVT, it prepares a quantum state whose measurement produces samples from a probability distribution close to the optimized distribution.

A natural approach to dequantization would, therefore, be to apply existing quantum-inspired techniques to this QSVT-based procedure. However, existing quantum-inspired classical algorithms do not apply directly to the matrix in (1). Although $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^{\top}$ has rank at most the number of training samples, the classical access to this matrix or its factors required by existing dequantization frameworks is not supplied by the original quantum input model [20–22]. Consequently, these frameworks cannot be applied directly to obtain an efficient classical sampler under the original input assumptions.

1.2 Main Contributions

Our two technical contributions are a classical sampler and an improved quantum sampler for the optimized probability distribution. The results of this paper are summarized in Table 1.

A classical algorithm for sampling from the optimized distribution. Our main result is a randomized classical algorithm for sampling from the optimized probability distribution used in the ridgelet-based sparse-subnetwork construction in [11]. Given the dataset and the regularization parameter λ, our algorithm outputs a hidden-node parameter by sampling from the optimized distribution determined by the regularized ridgelet coefficients while avoiding the above obstacles by explicitly exploiting the structure of the discrete ridgelet transform. As summarized in Table 1, this gives a polynomial-time classical sampler in the regime where the number of training samples $M = \mathrm{poly}(D)$ and the other relevant parameters are polynomially bounded.

An improved quantum algorithm for sampling from the optimized distribution. The technique used to construct our classical algorithm also improves the quantum algorithm for this sampling. In the quantum algorithm in [11], the matrix inversion with regularization in (1) is implemented with a runtime depending on the regularization parameter $1/\lambda$. Our analysis shows that this generic inverse step is not necessary for this sampling task. Therefore, we eliminate the $1/\lambda$ dependence in the runtime of our classical algorithm and the corresponding improved quantum algorithm.

Table 1: Comparison of approaches for sampling from the optimized ridgelet distribution.

Approach	End-to-end polytime	Preprocessing	M dependence	$1/\lambda$ dependence
Naive classical	No	None	–	Yes
Previous quantum [11]	Yes	$\widetilde{O}(M)$	$\mathrm{polylog}(M)$	Yes
This work, classical	Yes	$\widetilde{O}(M)$	M^2	No
This work, improved quantum	Yes	$\widetilde{O}(M)$	$\mathrm{polylog}(M)$	No

Numerical experiments. We numerically demonstrate the performance of our classical algorithm; after sampling hidden nodes, we train the output weights on these nodes. The resulting sparse networks achieve empirical risk comparable to exact sampling from the optimized distribution and substantially better than sampling from the uniform distribution (Fig. 1). We also numerically verify that our classical algorithm achieves polynomial runtime scaling, while the naive implementation of exact sampling from the same optimized distribution requires exponential time (Fig. 2).

Implications. Our results open an efficient classical route to finding a sparse network on conventional hardware via optimized sampling of trainable hidden nodes, without directly handling the large-scale dense network or relying on quantum hardware. More broadly, the techniques that we

develop dequantize a QML subroutine by exploiting problem-specific structure beyond the assumptions used in existing dequantization techniques, providing complementary tools for assessing when apparent quantum advantages in QML can be reproduced by efficient classical algorithms.

2 Settings

In this section, we define the learning setting used in this work. First, we define a ridgelet representation for a single-hidden-layer neural network over the discretized domain. Then, we introduce the empirical risk minimization and its optimal solution based on the representation. Finally, we define a probability distribution optimized for constructing a sparse subnetwork and clarify the sampling problem we address in this work. Appendix A summarizes basic notions and notations of quantum computation to describe our results, referring to the standard textbook [18] for more details. Following the bra-ket notation, we may represent a vector as a ket.

2.1 Discrete Ridgelet Transform

In this work, we consider a supervised regression problem. Let D be an input dimension, and $f : \mathbb{R}^D \to \mathbb{R}$ be an unknown target function to be learned. Let $g : \mathbb{R} \to \mathbb{R}$ denote an activation function such as rectified linear unit (ReLU) and sigmoid. A single-hidden-layer fully connected neural network approximates the target function f in the form $f(\boldsymbol{x}) \approx \sum_{n=1}^{N} w_n g(\boldsymbol{a}_n^\top \boldsymbol{x} - b_n)$, where N is the number of hidden-layer nodes, and w_n is a weight associated with node for $(\boldsymbol{a}_n, b_n)$.

Our goal is not to optimize all hidden nodes directly, but to construct a sparse subnetwork with a small number of hidden nodes that strongly affect the approximation. In the over-parameterized limit $N \to \infty$, a one-hidden-layer neural network is simplified into an integral representation $\mathcal{S}[w](\boldsymbol{x}) = \int_{\mathbb{R}^D \times \mathbb{R}} d(\boldsymbol{a}) db w(\boldsymbol{a}, b) g(\boldsymbol{a}^\top \boldsymbol{x} - b)$, and by using a function $r : \mathbb{R} \to \mathbb{R}$, we define a ridgelet transform $\mathcal{R}[f](\boldsymbol{a}, b) = \int_{\mathbb{R}^D} d(\boldsymbol{x}) f(\boldsymbol{x}) r(\boldsymbol{a}^\top \boldsymbol{x} - b)$ [10, 23–25]. Under a suitable admissibility condition on g and r, the reconstruction formula $\mathcal{S}[\mathcal{R}[f]] = f$ holds [10], and thus, we can consider $\mathcal{R}[f](\boldsymbol{a}, b)$ as a coefficient contributing to the evaluation of the neural network for each node parameterized by $(\boldsymbol{a}, b)$.

In this work, we perform analysis over discretized models rather than continuous models. In actual computation, inputs and parameters are represented with finite precision, e.g., one uses a bitstring to represent a real number. Consistent with this viewpoint, the formulation of the quantum ridgelet transform in previous research is also based on a discrete setting using a finite field [11]. Therefore, we use a discrete analog of this representation over $\mathbb{Z}_P \coloneqq \{0, 1, \ldots, P-1\}$, where P is a prime number, to compare classical and quantum algorithms for the same sampling task. In the following, when we write a sum, $\boldsymbol{x}$ and $\boldsymbol{a}$ run over $\mathbb{Z}_P^D$, and y and b over $\mathbb{Z}_P$ unless specified otherwise.

Discrete ridgelet representation: We suppose both an activation and a ridgelet function satisfy the following conditions over the domain $\mathbb{Z}_P$ instead of $\mathbb{R}$: (1) an activation function $g : \mathbb{Z}_P \to \mathbb{R}$ is assumed to be normalized as $\sum_b g(b) = 0$, $\|g\|_2^2 = \sum_b |g(b)|^2 = 1$, (2) we can choose r corresponding to g, satisfying a certain admissibility condition [10, 11], but, in this work, we choose a ridgelet function $r : \mathbb{Z}_P \to \mathbb{R}$ as $r = g$ for simplicity of notation.

In the above condition, by representing the integral over the continuous space $\mathbb{R}$ by the summation over the finite field $\mathbb{Z}_P$, the discretized neural network for $w : \mathbb{Z}_P^D \times \mathbb{Z}_P \to \mathbb{R}$ and the discrete ridgelet transform of $f : \mathbb{Z}_P^D \to \mathbb{R}$ are defined as, respectively, $\mathcal{S}[w](\boldsymbol{x}) \coloneqq P^{-\frac{D}{2}} \sum_{\boldsymbol{a},b} w(\boldsymbol{a}, b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P)$, and $\mathcal{R}[f](\boldsymbol{a}, b) \coloneqq P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) r((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P)$, where the coefficient $P^{-\frac{D}{2}}$ works as a normalization constant. The discretization of the neural network and the ridgelet transform preserves the reconstruction formula $f(\boldsymbol{x}) = \mathcal{S}[\mathcal{R}[f]](\boldsymbol{x})$. In particular, the ridgelet coefficients $\mathcal{R}[f]$ provide an exact representation of the target function in the discretized single-hidden-layer network [11]. It is convenient to represent the discrete ridgelet transform as a matrix $\boldsymbol{R} \coloneqq P^{-D/2} \sum_{\boldsymbol{x},\boldsymbol{a},b} r((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) |\boldsymbol{a}, b\rangle \langle \boldsymbol{x}|$. Then, for any function $f : \mathbb{Z}_P^D \to \mathbb{R}$, if we write $|f\rangle \coloneqq \sum_{\boldsymbol{x}} f(\boldsymbol{x}) |\boldsymbol{x}\rangle$, we have $\boldsymbol{R} |f\rangle = \sum_{\boldsymbol{a},b} \mathcal{R}[f](\boldsymbol{a}, b) |\boldsymbol{a}, b\rangle$.

2.2 Regularized Ridgelet Estimator and Optimized Distribution

Data and empirical objective. We consider a supervised regression task on the discrete input domain over $\mathbb{Z}_P^D$. Let $f : \mathbb{Z}_P^D \to \mathbb{R}$ be an unknown target function to be learned, and

let $p_{\mathrm{data}} : \mathbb{Z}_P^D \to \mathbb{R}_{\geq 0}$ be the input distribution. We are given a dataset of M input-output pairs $\{(\boldsymbol{x}_m, y_m)\}_{m\in[M]} \subseteq \mathbb{Z}_P^D \times \mathbb{R}$, where each input $\boldsymbol{x}_m$ is drawn from p_{data} and each label is given by $y_m = f(\boldsymbol{x}_m)$. The empirical distribution induced by the dataset is $\hat{p}_{\mathrm{data}} \coloneqq (1/M)\sum_{m\in[M]} \mathbf{1}\{\boldsymbol{x}_m = \boldsymbol{x}\}$. We aim to approximate f by a single-hidden-layer discretized neural network $\hat{f} = \mathcal{S}[w]$ from the training dataset. We measure the fit of $\hat{f}$ by the weighted empirical squared loss $J(w) \coloneqq \sum_{\boldsymbol{x}\in\mathbb{Z}_P^D} \hat{p}_{\mathrm{data}}(\boldsymbol{x})|f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})|^2$. To obtain a stable coefficient vector, we add the ℓ_2 penalty $\Omega(w) = \|P^{-\frac{D}{2}} w\|_2^2 = \sum_{\boldsymbol{a},b} |P^{-\frac{D}{2}} w(\boldsymbol{a}, b)|^2$ with a hyperparameter λ, and we consider a regularized least-squares regression problem, i.e., finding the optimal estimator w_λ^* satisfying

$$w_\lambda^* \coloneqq \arg\min_w \{\tilde{J}(w)\}, \tag{2}$$

where $\tilde{J}(w) \coloneqq J(w) + \lambda\Omega(w)$. The regularization makes the objective function strongly convex in the coefficient vector and fixes a unique minimizer. In particular, choosing $\lambda = O(\mathrm{poly}(\epsilon))$ makes the bias term $O(\epsilon)$, which guarantees the final empirical approximation guarantee $J(w) = O(\epsilon)$. We also define

$$\gamma \coloneqq \|P^{-\frac{D}{2}} w_\lambda^*\|_2^2 \tag{3}$$

that measures the scale of the normalized regularized ridgelet coefficients and appears in the runtime bounds of our sampling algorithms.

Optimized Probability Distribution for Sparse-Subnetwork Construction. The optimal estimator w_λ^* in (2) determines the contribution of each node parameterized by $(\boldsymbol{a}, b)$ in the discretized ridgelet representation. To construct a sparse subnetwork, we would like to sample hidden-node parameters with large normalized ridgelet weights. Following [11], we define the optimized probability distribution as follows.

Definition 1 (Optimized probability distribution)**.** *For parameters $\lambda > 0$ and $\Delta > 0$, the optimized probability distribution is defined by*

$$p_{\lambda,\Delta}^*(\boldsymbol{a}, b) \coloneqq \frac{1}{Z} \frac{\left|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a}, b)\right|^2}{\left|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a}, b)\right|^2 + \Delta} \tag{4}$$

where Z is a normalization constant ensuring that $p_{\lambda,\Delta}^*$ is a probability distribution, i.e., $\sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P} p_{\lambda,\Delta}^*(\boldsymbol{a}, b) = 1$.

This probability distribution assigns large probability masses to high-weight parameters. The smoothing parameter Δ smooths the distribution by saturating the high-weight nodes and more strongly suppressing the low-weight nodes: we can choose the probability distribution proportional to the square of the optimal weights w_λ^*, but this choice may concentrate too much probability mass on a few high-weight nodes, leading to repeated sampling of the same nodes.

Equivalently, using the matrix expression of w_λ^*, we write the quantum state

$$\left|p_{\lambda,\Delta}^*\right\rangle \propto \left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-1/2} \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1} \boldsymbol{R}\left|\psi_{\mathrm{in}}\right\rangle, \tag{5}$$

where $\boldsymbol{W}_\lambda \coloneqq \frac{1}{\gamma}\sum_{\boldsymbol{a},b} |P^{-D/2} w_\lambda^*(\boldsymbol{a}, b)|^2 \,|\boldsymbol{a}, b\rangle\langle\boldsymbol{a}, b|$, $\hat{\boldsymbol{P}}_{\mathrm{data}} \coloneqq \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})\,|\boldsymbol{x}\rangle\langle\boldsymbol{x}|$, and $|\psi_{\mathrm{in}}\rangle \coloneqq \hat{\boldsymbol{P}}_{\mathrm{data}}\,|f\rangle$. This form is used in the quantum ridgelet transform algorithm [11], i.e., the quantum algorithm prepares a quantum state close to (5) and measures it to sample from the optimized probability distribution.

The computational task studied in this paper is to sample from a distribution close to $p_{\lambda,\Delta}^*$ in total variation distance. More precisely, given $\lambda > 0$, $\Delta > 0$, and an accuracy parameter $\delta \in (0, 1)$, our goal is to output one sample from a distribution $\mathcal{D}'$ over $\mathbb{Z}_P^D \times \mathbb{Z}_P$ such that $d_{\mathrm{TV}}(p_{\lambda,\Delta}^*, \mathcal{D}') \leq \delta$ in total variation distance.

Once such a sampling algorithm is obtained, repeated sampling yields hidden-node parameters for a sparse single-hidden-layer network. The end-to-end $O(\epsilon)$ empirical-risk guarantee follows from the bounds for finding the lottery-ticket theorem in [11], under the assumption $|P^{-D/2} w_\lambda^*(\boldsymbol{a}_j, b_j)| \leq \alpha j^{-(1+\beta)}$, where the coefficients are ordered by nonincreasing magnitude and $\alpha, \beta > 0$ are independent of D.

3 Sampling from Optimized Probability Distribution

We now provide our main results. Our primary result is a randomized classical algorithm for sampling from the optimized probability distribution $p^*_{\lambda,\Delta}$. We explain the two main ideas behind the sampling algorithm. Finally, we state the corresponding improvement of the quantum sampling algorithm.

3.1 Classical sampling algorithm for optimized probability distribution

Our main result for the classical sampling algorithm is as follows.

Theorem 2 (Classical sampler for the optimized distribution)**.** *Given the D-dimensional dataset $\{\boldsymbol{x}_m, y_m\}_{m\in[M]}$, and parameters $\lambda, \Delta > 0$ with $\gamma > 0$ in (3), for any accuracy parameter $\delta \in (0,1)$, there exists a randomized classical algorithm that outputs one sample from a distribution $\mathcal{D}'$ whose total variation distance from $p^*_{\lambda,\Delta}$ in Definition 1 is at most δ, with sample runtime*

$$O(D\log(1/\delta)\mathrm{polylog}(P)) \times \tilde{O}(M^2\,(1+\gamma/\Delta)). \tag{6}$$

In particular, if $M = \mathrm{poly}(D)$ and the relevant parameters γ, Δ^{-1}, $\log(1/\delta)$, and $\log P$ are polynomially bounded in D, the optimized-distribution sampling task admits a polynomial-time randomized classical implementation. See Appendix C for the proof. We explain the main ideas below.

3.2 Main Idea of Classical Sampling Algorithm

The difficulties in this sampling task lie in the QSVT for the high-dimensional matrix in (5). A natural approach would be to classically simulate the QSVT part by using quantum-inspired techniques. Our target matrix to be inverted, however, does not satisfy the assumptions required for such approaches: the original quantum input model does not supply the classical access to the relevant matrices required by these approaches. Our algorithm instead uses a problem-specific structure together with a quantum-inspired sampling-and-query data structure.

Avoidance of the dense inverse. The central observation of our algorithm is that the apparent high-dimensional inverse is not a general inverse. Although $(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I})^{-1}$ acts on the high-dimensional hidden-node spaces, it has a special ridgelet structure, i.e., $\boldsymbol{R}$ is an isometry and $\hat{\boldsymbol{P}}_{\mathrm{data}}$ is diagonal. Thus, the Woodbury identity [26, 27] yields the following decomposition:

$$\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1} = \boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}^\top + \frac{1}{\lambda P^{-D}}\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top\right). \tag{7}$$

The first term acts on the image of $\boldsymbol{R}$, while the second term acts on its orthogonal complement. Since the vector to which the inverse is applied is $\boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle$, the orthogonal-complement term vanishes $\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top\right)\boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle = 0$. Consequently, the regularized ridgelet coefficient vector can be written as

$$w^*_\lambda = \boldsymbol{R}\,|\phi\rangle\,, \tag{8}$$

where $|\phi\rangle = (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I})^{-1}\,|\psi_{\mathrm{in}}\rangle$. Thus, the dense inverse over the hidden-node parameter space is reduced to an entrywise diagonal reweighting over the data domain, followed by the ridgelet transform. This reduction, which exploits the ridgelet-specific settings, is the key reason why our algorithm avoids implementing the regularized inverse as a general high-dimensional matrix transformation.

After the action of $\boldsymbol{R}$, each amplitude becomes a global sum involving all coordinates of the transformed vector weighted by the ridgelet function r. In this sense, the computational bottleneck is a highly complex correlation sum induced by the ridgelet transform rather than the local access to individual entries in $|\phi\rangle$.

This characteristic is reminiscent of the Forrelation problem, where one is asked to estimate the correlation between one Boolean function and the Fourier transform of another [28]. That problem exhibits an optimal quantum-classical query separation: it can be solved with a constant number of quantum queries, while any randomized classical algorithm requires an exponentially large number of queries in the oracle setting. Nevertheless, in this work, we avoid this problem by using the sparsity M on the vector $|\phi\rangle$, i.e., the correlation in the amplitude is reduced to a summation of M terms

$$\sum_{\boldsymbol{x}} r(\boldsymbol{a}^\top\boldsymbol{x} - b \bmod P)\frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x})f(\boldsymbol{x})}{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) + \lambda P^{-D}} = \sum_{\boldsymbol{x}\in\mathcal{X}_{\mathrm{data}}} r(\boldsymbol{a}^\top\boldsymbol{x} - b \bmod P)\frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x})f(\boldsymbol{x})}{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) + \lambda P^{-D}}, \tag{9}$$

where $\mathcal{X}_{\mathrm{data}} \coloneqq \mathrm{Supp}(\hat{p}_{\mathrm{data}})$ is a support of the empirical probability distribution.

From weight evaluation to sampling. This reduction gives an efficient way to evaluate the unnormalized weight of the optimized probability distribution $p^*_{\lambda,\Delta}$ for each hidden-node parameter $(\boldsymbol{a}, b)$. However, pointwise evaluation is not enough to sample from $p^*_{\lambda,\Delta}$. In fact, for sampling, the unnormalized weights need to be converted into probabilities. Thus, a direct sampler requires either computing Z in Definition 1 or avoiding this normalization step. Moreover, a naive rejection sampling based on the uniform proposal distribution can be inefficient, since the optimized distribution may concentrate on a small subset of high-weight hidden nodes.

To address these problems, we design a proposal distribution suitable for the optimized coefficients. Let $K \coloneqq |\mathcal{X}_{\mathrm{data}}| \leq M$. We define

$$q(\boldsymbol{a}, b) \coloneqq \frac{1}{\gamma} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{data}}} \left| P^{-D/2} \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \right|^2 \left| P^{-D/2} r\left(\boldsymbol{a}^\top \boldsymbol{x} - b \mod P\right) \right|^2, \tag{10}$$

which is chosen so that every high-weight hidden node with a large optimized weight is proposed with sufficiently large probability.

This proposal distribution q is efficiently samplable. To sample from q, we use a quantum-inspired data structure for the vector $|\phi\rangle$. This is the same type of binary-tree data structure used for efficient quantum state preparation and quantum-inspired classical algorithms [20, 21, 29, 30]. In fact, the first factor in each summand is proportional to the squared magnitude of $|\phi\rangle$, while the second factor is the probability of a ridgelet transform compatible with the sampled data point. Thus, to sample from q, we first sample a data point $\boldsymbol{x}$ according to the squared entries of $|\phi\rangle$. Then, we sample $\boldsymbol{a}$ uniformly from $\mathbb{Z}_P^D$ and sample $t \in \mathbb{Z}_P$ with probability $|r(t)|^2$. Finally, we set $b = \boldsymbol{a}^\top \boldsymbol{x} - t \bmod P$. This procedure produces $(\boldsymbol{a}, b)$ with probability $q(\boldsymbol{a}, b)$. Therefore, the proposal distribution q is not only analytically useful for rejection sampling, but also efficiently samplable.

Given a proposal sample $(\boldsymbol{a}, b) \sim q$, we accept it with probability

$$\frac{\Delta}{\Delta + \left| P^{-D/2} w^*_\lambda(\boldsymbol{a}, b) \right|^2} \frac{\left| P^{-D/2} w^*_\lambda(\boldsymbol{a}, b) \right|^2}{K \gamma q(\boldsymbol{a}, b)}. \tag{11}$$

The Cauchy-Schwarz inequality guarantees that this acceptance probability is at most one. Conditioned on acceptance, the output distribution is exactly $p^*_{\lambda,\Delta}$. Repeating this rejection-sampling procedure sufficiently many times makes the failure probability at most δ. This yields the randomized classical sampler in Theorem 2.

3.3 Improved quantum sampling algorithm

The same inverse reduction also improves the quantum sampling algorithm of [11]. The original quantum algorithm prepares the sampling state in (5) by implementing the regularized inverse $(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I})^{-1}$ as a high-dimensional matrix transformation. This implementation leads to an explicit $1/\lambda$ dependence in the runtime.

Our inverse reduction shows that this inverse step is unnecessary. In fact, by (8), the coefficient quantum state can be prepared by preparing $|\phi\rangle$ and then applying the quantum ridgelet transform $\boldsymbol{R}$. Importantly, the $|\phi\rangle$ used here is exactly the same data-dependent vector for which our classical algorithm constructs a data structure. The same structure also provides the quantum state preparation access required by the quantum algorithm. Therefore, the quantum algorithm no longer needs to implement the regularized inverse over the hidden-node space as a QSVT transformation. The remaining QSVT step is only the smoothing transformation associated with $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$ as in the original sampling procedure. This yields the following improved quantum sampling algorithm. The proof is given in Appendix D.

Theorem 3 (Improved quantum sampler for the optimized distribution)**.** *Given the D-dimensional dataset $\{\boldsymbol{x}_m, y_m\}_{m\in[M]}$ and parameters $\lambda, \Delta > 0$, with γ defined in* (3)*, for any sampling precision $\delta \in (0, 1)$, there exists a quantum algorithm that outputs one sample from a distribution $\mathcal{D}'$ whose total variation distance from $p^*_{\lambda,\Delta}$ in Definition 1 is at most δ, with sample runtime*

$$\tilde{O}(D(1 + \gamma/\Delta)\mathrm{polylog}(MP/\delta)) \tag{12}$$

after preprocessing time $\tilde{O}(M)$.

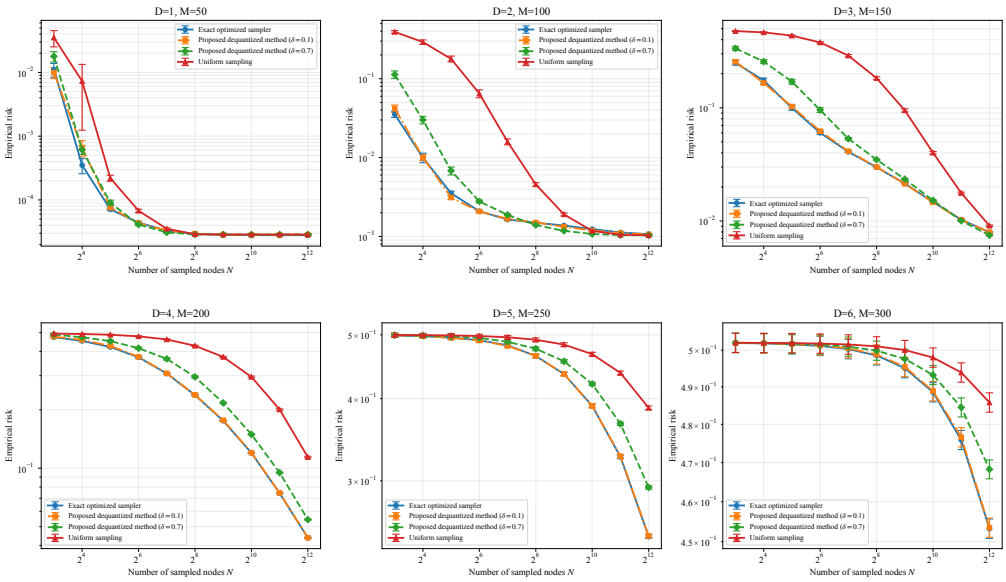

Figure 1: Empirical risk of sparse subnetworks constructed from N sampled hidden nodes. We compare exact sampling from the optimized distribution (blue solid line), our dequantized classical algorithm with $\delta = 0.1$ (orange dashdot line), $\delta = 0.7$ (green dashed line), and sampling from the non-optimized uniform distribution (red solid line). The experiments use $N \in \{8, 16, \ldots, 4096\}$. Curves show the mean over 50 executions, while error bars are the standard error of the mean.

4 Numerical Experiments

In this section, we numerically demonstrate that our algorithm efficiently yields a desired sparse subnetwork. The proposed sampling algorithms construct the sparse subnetwork by sampling hidden nodes from the optimized probability distribution; i.e., the sampled nodes define the basis functions for the subnetwork. After sampling these important nodes, we train the output weight on the nodes. The previous algorithm in [11] for this sampling relied on quantum computation; in contrast, our dequantized sampler in Theorem 2 is fully classical and can be executed on a conventional computer.

We numerically evaluate whether the sampled nodes generated by our sampler yield a lower empirical risk than those drawn from the non-optimized uniform distribution. Moreover, we evaluate its runtime scaling to verify that the proposed algorithm avoids the exponential runtime in the data dimension required by a naive classical implementation of sampling from the optimized distribution.

Empirical-risk comparison. For the empirical-risk experiment, we work on the finite domain $\mathbb{Z}_P^D$ with $P = 7$ and $D \in \{1, \ldots, 6\}$. This dimension range is chosen to keep the exact optimized sampler computationally feasible, since it explicitly enumerates all P^{D+1} hidden-node parameters. We set the function to be learned as a sine function. For each D, we draw $M = 50D$ training inputs independently from the uniform distribution $p_{\mathrm{data}}(\boldsymbol{x}) = 1/P^D$, and the empirical distribution $\hat{p}_{\mathrm{data}}$ is induced by the training dataset. We use ReLU as the activation function, and the ridgelet function $g = r$. We set $\lambda P^{-D} = 10^{-3}$, and set $\Delta = \gamma$ for efficiency. Then, we compare exact sampling from the optimized distribution $p^*_{\lambda,\Delta}$ in Definition 1, our sampler approximately sampling from $p^*_{\lambda,\Delta}$ with accuracy parameters $\delta = 0.1$ and $\delta = 0.7$, and sampling from the uniform distribution. Given N sampled hidden nodes $\{(\boldsymbol{a}_n, b_n)\}_{n=1}^N$, we fit the weights by ridge regression and evaluate the empirical risk $\sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) |f_D(\boldsymbol{x}) - \hat{f}_N(\boldsymbol{x})|^2$, where $\hat{f}_N$ is a trained subnetwork constructed from N sampled nodes. This ridge estimator is obtained by solving the corresponding normal equations using linear algebra routines in NumPy. We plot the achievable empirical risk with 50 repetitions for each procedure.

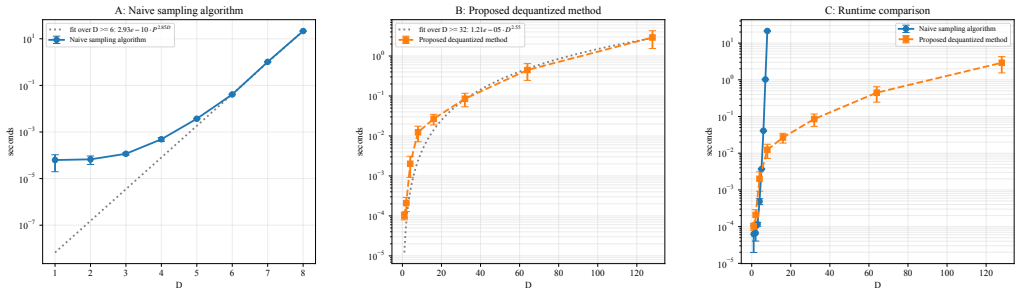

Figure 2: Runtime comparison between the naive sampling algorithm (blue solid line) and our dequantized classical algorithm (orange dashed line) in Theorem 2. Each curve shows the mean over 20 executions, while error bars are 95% confidence intervals. Fitted curves are scaling guides.

Figure 1 shows that for all tested dimensions, our sampler closely follows the exact sampler from the optimized distribution. This implies that the proposed rejection sampling preserves the useful bias of the optimized distribution. In contrast, sampling from the non-optimized uniform distribution consistently requires more nodes to achieve the same empirical risk in the middle regime of N. For large dimension D, the rigorous setting $\delta = 0.1$ tends to track the exact sampler more closely. Thus, these numerical results support the use of our classical algorithm as a replacement for exact sampling from the optimized distribution in sparse-subnetwork construction.

Runtime comparison. For the runtime experiment, we compare the runtime of our dequantized sampler with a naive classical implementation of exact sampling based on directly computing the inverse of a dense matrix. We use the same data-generation procedure as the above empirical-risk experiment, but set $P = 3$ so that the naive algorithm can be executed up to moderate dimensions D beyond those in the empirical-risk experiment. For $D \in \{1, \ldots, 8\}$, the naive algorithm explicitly solves the system of linear equations over the hidden-node-parameter space before sampling. By contrast, our sampler is tested for $D \in \{1, 2, 4, 8, 16, 32, 64, 128\}$.

In Figure 2, the naive algorithm rapidly becomes computationally expensive as D increases, with its runtime growing exponentially in D; in particular, for $D \geq 6$, the fitted curve scales as $P^{2.85D}$ in the measured range. This behavior is consistent with the cost of solving the hidden-space linear system; i.e., the number of hidden-node parameters is P^{D+1}, and a dense linear solve over this space costs $O((P^{D+1})^3) = O(P^{3D})$. In contrast, our sampler remains executable up to larger dimensions than the naive algorithm, and exhibits substantially milder growth; in particular, the fitted curve for our sampler scales as $D^{2.55}$ in the measured range. This is consistent with the theoretical runtime bound in Theorem 2; that is, the dominant dimension-dependent runtime factor of our algorithm is DM^2 in Theorem 2, and in this experiment, we set $M = 50D$, leading to the expected scale $DM^2 = O(D^3)$. Thus, the runtime experiment numerically supports our runtime analysis: the data-dimension scaling is reduced from exponential to polynomial by our dequantized classical algorithm.

5 Conclusion and Discussion

We have developed a quantum-inspired classical algorithm for sampling from the optimized probability distribution used for sparse-subnetwork construction, motivated by the search for lottery tickets in dense neural networks. In our classical algorithm, we exploit the isometry structure of the discrete ridgelet transform to avoid inverting dense matrices over the exponentially large-dimensional space by reducing it to the inversion of a diagonal matrix supported on empirical data. This yields a polynomial-time classical algorithm when problem parameters are polynomially bounded. The same technique also improves the previous quantum algorithm in [11] for this sampling by removing this matrix-inversion requirement. Numerical experiments show that the proposed classical sampler can construct sparse subnetworks with empirical risk comparable to exact sampling from the optimized distribution, and substantially better than sampling from the uniform distribution. The runtime experiments further support the theoretical runtime improvement over the previous classical approach.

These results provide a classical alternative to the previous quantum approach for sparse-subnetwork construction, underscoring the importance of comparing these classical and improved quantum algorithms under plausible future hardware assumptions and hardware-specific overhead estimates.

Acknowledgments and Disclosure of Funding

NI was supported by JST BOOST, Japan Grant Number JPMJBS2418, JST CREST Grant Number JPMJCR25I5, JSPS KAKENHI Grant Number JP23K21643, and MEXT Quantum Leap Flagship Program (MEXT QLEAP) JPMXS0118069605, JPMXS0120351339. HY was supported by JST PRESTO Grant Number JPMJPR201A, JPMJPR23FC, JSPS KAKENHI Grant Number JP23K19970, JST CREST Grant Number JPMJCR25I5, JST [Moonshot R&D] [Grant Number JPMJMS256J], and Faculty Research Funding from Google Quantum AI. SS was supported by JST BOOST JPMJBY24E2, JST CREST Grant Number JPMJCR25I5, and JSPS KAKENHI JP24K21316. MM was supported by MEXT Quantum Leap Flagship Program (MEXT QLEAP) JPMXS0118069605, JPMXS0120351339, JST CREST Grant Number JPMJCR25I5, JST ASPIRE Grant Number JPMJAP25A3, JSPS KAKENHI Grant Number JP23K21643, JST NEXUS Grant Number JPMJNX26C9, and IBM Quantum.

The authors declare no additional competing interests beyond the funding and affiliations disclosed above.

References

[1] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. URL https://arxiv.org/abs/1510.00149.

[2] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

[3] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[4] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[6] Bohan Liu, Zijie Zhang, Peixiong He, Zhensen Wang, Yang Xiao, Ruimeng Ye, Yang Zhou, Wei-Shinn Ku, and Bo Hui. A survey of lottery ticket hypothesis, 2024. URL https://arxiv.org/abs/2403.04861.

[7] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

[8] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf.

[9] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research*, 18(21):1–38, 2017. URL http://jmlr.org/papers/v18/15-178.html.

[10] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2015.12.005. URL https://www.sciencedirect.com/science/article/pii/S1063520315001748.

[11] Hayata Yamasaki, Sathyawageeswar Subramanian, Satoshi Hayakawa, and Sho Sonoda. Quantum ridgelet transform: Winning lottery ticket of neural networks with quantum computation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39008–39034. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yamasaki23a.html.

[12] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Elsevier, 2014. URL https://www.sciencedirect.com/book/9780128009536/quantum-machine-learning.

[13] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, Sep 2017. ISSN 1476-4687. doi: 10.1038/nature23474. URL https://doi.org/10.1038/nature23474.

[14] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi: 10.1103/PhysRevLett.103.150502. URL https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[15] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 193–204, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367059. doi: 10.1145/3313276.3316366. URL https://doi.org/10.1145/3313276.3316366.

[16] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. ISSN 2521-327X. doi: 10.22331/q-2018-08-06-79. URL https://doi.org/10.22331/q-2018-08-06-79.

[17] Rajeev Acharya, Dmitry A. Abanin, Laleh Aghababaie-Beni, Igor Aleiner, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Nikita Astrakhantsev, Juan Atalaya, Ryan Babbush, Dave Bacon, Brian Ballard, Joseph C. Bardin, Johannes Bausch, Andreas Bengtsson, Alexander Bilmes, Sam Blackwell, Sergio Boixo, Gina Bortoli, Alexandre Bourassa, Jenna Bovaird, Leon Brill, Michael Broughton, David A. Browne, Brett Buchea, Bob B. Buckley, David A. Buell, Tim Burger, Brian Burkett, Nicholas Bushnell, Anthony Cabrera, Juan Campero, Hung-Shen Chang, Yu Chen, Zijun Chen, Ben Chiaro, Desmond Chik, Charina Chou, Jahan Claes, Agnetta Y. Cleland, Josh Cogan, Roberto Collins, Paul Conner, William Courtney, Alexander L. Crook, Ben Curtin, Sayan Das, Alex Davies, Laura De Lorenzo, Dripto M. Debroy, Sean Demura, Michel Devoret, Agustin Di Paolo, Paul Donohoe, Ilya Drozdov, Andrew Dunsworth, Clint Earle, Thomas Edlich, Alec Eickbusch, Aviv Moshe Elbag, Mahmoud Elzouka, Catherine Erickson, Lara Faoro, Edward Farhi, Vinicius S. Ferreira, Leslie Flores Burgos, Ebrahim Forati, Austin G. Fowler, Brooks Foxen, Suhas Ganjam, Gonzalo Garcia, Robert Gasca, Élie Genois, William Giang, Craig Gidney, Dar Gilboa, Raja Gosula, Alejandro Grajales Dau, Dietrich Graumann, Alex Greene, Jonathan A. Gross, Steve Habegger, John Hall, Michael C. Hamilton, Monica Hansen, Matthew P. Harrigan, Sean D. Harrington, Francisco J. H. Heras, Stephen Heslin, Paula Heu, Oscar Higgott, Gordon Hill, Jeremy Hilton, George Holland, Sabrina Hong, Hsin-Yuan Huang, Ashley Huff, William J. Huggins, Lev B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Stephen Jordan, Chaitali Joshi, Pavol Juhas, Dvir Kafri, Hui Kang, Amir H. Karamlou, Kostyantyn Kechedzhi, Julian Kelly, Trupti Khaire, Tanuj Khattar, Mostafa Khezri, Seon Kim, Paul V. Klimov, Andrey R. Klots, Bryce Kobrin, Pushmeet Kohli, Alexander N. Korotkov, Fedor Kostritsa, Robin Kothari, Borislav Kozlovskii, John Mark Kreikebaum, Vladislav D. Kurilovich, Nathan Lacroix, David Landhuis, Tiano Lange-Dei, Brandon W. Langley, Pavel Laptev, Kim-Ming Lau, Loïck Le Guevel, Justin Ledford, Joonho Lee, Kenny Lee, Yuri D. Lensky, Shannon Leon, Brian J. Lester, Wing Yan Li, Yin Li, Alexander T. Lill, Wayne Liu, William P. Livingston, Aditya Locharla, Erik Lucero, Daniel Lundahl, Aaron Lunt, Sid Madhuk, Fionn D. Malone, Ashley Maloney, Salvatore Mandrà, James Manyika, Leigh S. Martin, Orion Martin, Steven Martin, Cameron Maxfield, Jarrod R. McClean, Matt McEwen, Seneca Meeks, Anthony Megrant, Xiao Mi, Kevin C. Miao, Amanda Mieszala, Reza Molavi, Sebastian Molina, Shirin Montazeri, Alexis Morvan, Ramis Movassagh, Wojciech Mruczkiewicz, Ofer Naaman,

Matthew Neeley, Charles Neill, Ani Nersisyan, Hartmut Neven, Michael Newman, Jiun How Ng, Anthony Nguyen, Murray Nguyen, Chia-Hung Ni, Murphy Yuezhen Niu, Thomas E. O'Brien, William D. Oliver, Alex Opremcak, Kristoffer Ottosson, Andre Petukhov, Alex Pizzuto, John Platt, Rebecca Potter, Orion Pritchard, Leonid P. Pryadko, Chris Quintana, Ganesh Ramachandran, Matthew J. Reagor, John Redding, David M. Rhodes, Gabrielle Roberts, Eliott Rosenberg, Emma Rosenfeld, Pedram Roushan, Nicholas C. Rubin, Negar Saei, Daniel Sank, Kannan Sankaragomathi, Kevin J. Satzinger, Henry F. Schurkus, Christopher Schuster, Andrew W. Senior, Michael J. Shearn, Aaron Shorter, Noah Shutty, Vladimir Shvarts, Shraddha Singh, Volodymyr Sivak, Jindra Skruzny, Spencer Small, Vadim Smelyanskiy, W. Clarke Smith, Rolando D. Somma, Sofia Springer, George Sterling, Doug Strain, Jordan Suchard, Aaron Szasz, Alex Sztein, Douglas Thor, Alfredo Torres, M. Mert Torunbalci, Abeer Vaishnav, Justin Vargas, Sergey Vdovichev, Guifre Vidal, Benjamin Villalonga, Catherine Vollgraff Heidweiller, Steven Waltman, Shannon X. Wang, Brayden Ware, Kate Weber, Travis Weidel, Theodore White, Kristi Wong, Bryan W. K. Woo, Cheng Xing, Z. Jamie Yao, Ping Yeh, Bicheng Ying, Juhwan Yoo, Noureldin Yosri, Grayson Young, Adam Zalcman, Yaxing Zhang, Ningfeng Zhu, Nicholas Zobrist, Google Quantum AI, and Collaborators. Quantum error correction below the surface code threshold. *Nature*, 638(8052):920–926, Feb 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08449-y. URL https://doi.org/10.1038/s41586-024-08449-y.

[18] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 10th edition, 2011. ISBN 9781107002173. URL https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE.

[19] Sanjeev Arora and Boaz Barak. *Computational complexity: A Modern Approach*. Cambridge University Press, 2009. doi: 10.1017/CBO9780511804090.

[20] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 217–228, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367059. doi: 10.1145/3313276.3316310. URL https://doi.org/10.1145/3313276.3316310.

[21] Nai-Hui Chia, András Pal Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. *J. ACM*, 69(5), October 2022. ISSN 0004-5411. doi: 10.1145/3549524. URL https://doi.org/10.1145/3549524.

[22] François Le Gall. Robust dequantization of the quantum singular value transformation and quantum machine learning algorithms. *computational complexity*, 34(1):2, 2025. doi: 10.1007/s00037-024-00262-3. URL https://link.springer.com/article/10.1007/s00037-024-00262-3.

[23] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993. doi: 10.1109/18.256500.

[24] Noboru Murata. An integral representation of functions using three-layered networks and their approximation bounds. *Neural Networks*, 9(6):947–956, 1996. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(96)00000-7. URL https://www.sciencedirect.com/science/article/pii/0893608096000007.

[25] Emmanuel J. Candès. *Ridgelets: Theory and Applications*. Ph.d. thesis, Stanford University, 1998. URL https://candes.su.domains/publications/downloads/Thesis.pdf.

[26] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989. doi: 10.1137/1031049. URL https://doi.org/10.1137/1031049.

[27] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013. doi: 10.1137/1.9781421407944. URL https://epubs.siam.org/doi/abs/10.1137/1.9781421407944.

[28] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 307–316, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450335362. doi: 10.1145/2746539.2746547. URL https://doi.org/10.1145/2746539.2746547.

[29] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. URL https://arxiv.org/abs/quant-ph/0208112.

[30] Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-029-3. doi: 10.4230/LIPIcs.ITCS.2017.49. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2017.49.

[31] Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: hardness and applications to quantum chemistry and the quantum pcp conjecture. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 19–32, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392648. doi: 10.1145/3519935.3519991. URL https://doi.org/10.1145/3519935.3519991.

[32] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 636–647, Dagstuhl, Germany, 2012. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-939897-35-4. doi: 10.4230/LIPIcs.STACS.2012.636. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2012.636.

[33] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017. doi: 10.1137/16M1087072. URL https://doi.org/10.1137/16M1087072.

A Preliminaries

A.1 Notations and general definitions

Let $\mathbb{N}$ be the set of natural numbers. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. In this work, we abbreviate $\log_2$ to $\log$ for simplicity of notation. Let P be a prime number and $\mathbb{Z}_P \coloneqq \{0, 1, \ldots, P-1\}$, where all additions and multiplications in $\mathbb{Z}_P$ are considered modulo P. Unless otherwise specified, $\boldsymbol{x}$ and $\boldsymbol{a}$ range over $\mathbb{Z}_P^D$, while b ranges over $\mathbb{Z}_P$. For $\boldsymbol{a}, \boldsymbol{x} \in \mathbb{Z}_P^D$, we write the inner product as $\boldsymbol{a}^\top \boldsymbol{x} \coloneqq \sum_{i=1}^{D} a_i x_i \pmod P$, where a_i and x_i denote the i-th elements of $\boldsymbol{a}$ and $\boldsymbol{x}$, respectively. Let $\mathbb{R}$ and $\mathbb{C}$ be the sets of real numbers and complex numbers, respectively. For a vector $\boldsymbol{v} \in \mathbb{C}^n$, $\|\boldsymbol{v}\|_2$ and v_i denote the l_2-norm and the i-th entry of $\boldsymbol{v}$, respectively. For a matrix $\boldsymbol{A}$, let $\boldsymbol{A}^\dagger$ and $\boldsymbol{A}^\top$ denote the conjugate transpose and the transpose of $\boldsymbol{A}$, respectively. For a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$, let $\|\boldsymbol{A}\|_{\mathrm{op}}$ and $\|\boldsymbol{A}\|_{\mathrm{F}}$ denote the operator norm induced by the l_2-norm and the Frobenius norm, respectively. For a Hermitian matrix $\boldsymbol{A} \in \mathbb{C}^{n \times n}$, let $\lambda_{\max}(\boldsymbol{A})$ and $\lambda_{\min}(\boldsymbol{A})$ denote the maximum and minimum eigenvalues of $\boldsymbol{A}$, respectively.

We introduce basic quantum notations (see [18] for more introductory details). In quantum computation, we use standard bra-ket notation for finite-dimensional Hilbert spaces. For a finite set Ω, let $\mathcal{H}_\Omega \coloneqq \mathbb{C}^{|\Omega|}$. We write this basis as $\{|\omega\rangle \mid \omega \in \Omega\}$. Thus, a vector $\boldsymbol{v} \in \mathbb{C}^{|\Omega|}$ can be written as the ket

$$|\boldsymbol{v}\rangle \coloneqq \sum_{\omega \in \Omega} v(\omega) |\omega\rangle \in \mathcal{H}_\Omega. \tag{13}$$

This ket is not assumed to be normalized unless explicitly stated. For a ket $|\boldsymbol{v}\rangle$, its conjugate transpose is denoted by $\langle \boldsymbol{v}|$. For kets $|\boldsymbol{v}\rangle$ and $|\boldsymbol{u}\rangle$, their inner product is written as $\langle \boldsymbol{v}|\boldsymbol{u}\rangle$. For sets Ω, Θ representing the standard bases, an (ω, θ)-th entry of a matrix $\boldsymbol{A} \in \mathbb{C}^{|\Omega| \times |\Theta|}$ is written as $\langle \omega | \boldsymbol{A} | \theta \rangle$ for every $\omega \in \Omega$ and $\theta \in \Theta$.

As a conventional computation uses a bit $\{0, 1\}$, quantum computation uses a quantum bit (qubit) as the minimum computational element. A qubit is represented by $\mathbb{C}^2$, and a single-qubit state is a normalized vector in $\mathbb{C}^2$. With respect to the computational basis $\{|0\rangle, |1\rangle\}$, it has the form $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ for $\alpha_0, \alpha_1 \in \mathbb{C}$, where $|\alpha_0|^2 + |\alpha_1|^2 = 1$. An n-qubit register is described by the tensor-product space $(\mathbb{C}^2)^{\otimes n}$. Its computational basis can be indexed either by bitstrings in $\{0,1\}^n$ or by integers in $\{0, 1, \ldots, 2^n - 1\}$. Therefore, any n-qubit state can be written as $|\psi\rangle = \sum_{z=0}^{2^n-1} \alpha_z |z\rangle$ satisfying the normalized condition $\sum_{z}^{2^n-1} |\alpha_z|^2 = 1$, and we call α_z an amplitude. When two quantum registers are represented by Hilbert spaces $\mathcal{H}_\Omega$ and $\mathcal{H}_\Theta$, the composite system is represented by the tensor product $\mathcal{H}_\Omega \otimes \mathcal{H}_\Theta$. For some basis states, we use the abbreviation $|\omega, \theta\rangle \coloneqq |\omega\rangle \otimes |\theta\rangle$. More generally, for $\boldsymbol{v} = (v_1, \ldots v_D)$, we write $|\boldsymbol{v}\rangle \coloneqq |v_1\rangle \otimes \cdots \otimes |v_D\rangle$.

A transformation of a quantum system without measurement is represented by a unitary operator. If $|\psi\rangle \in \mathcal{H}$ is a quantum state and $\boldsymbol{U} : \mathcal{H} \to \mathcal{H}$ is unitary, then the state after the transformation is $\boldsymbol{U}|\psi\rangle$. The unitary condition $\boldsymbol{U}^\dagger \boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{I}$ ensures the normalization condition of quantum states, i.e.

$$\|\boldsymbol{U}|\psi\rangle\|_2^2 = \langle \psi | \boldsymbol{U}^\dagger \boldsymbol{U} | \psi \rangle = \langle \psi | \psi \rangle = 1. \tag{14}$$

Thus, a unitary operator maps quantum states to quantum states. We will also use isometry transformations, which allow distinct input and output Hilbert spaces with different dimensions. We say that a linear operator $\boldsymbol{V} : \mathcal{H}_1 \to \mathcal{H}_2$ is an isometry if $\boldsymbol{V}^\dagger \boldsymbol{V} = \boldsymbol{I}_{\mathcal{H}_1}$. Therefore, an isometry also maps quantum states to quantum states, although the state may be embedded in a larger Hilbert space. Although physical transformations of closed quantum systems are unitary, an isometry transformation can be implemented by a unitary transformation after adding auxiliary registers initialized in a fixed state, i.e., there exists a unitary operator $\boldsymbol{U}$ on a larger Hilbert space such that $\boldsymbol{U}(|\psi\rangle \otimes |0\rangle) = \boldsymbol{V}|\psi\rangle$ for all $|\psi\rangle$, considering $\mathcal{H}_2 \simeq \mathcal{H}_1 \otimes \mathcal{H}_{\mathrm{aux}}$.

A quantum measurement converts amplitudes into probability mass. If $|\psi\rangle = \sum_{\omega \in \Omega} \alpha_\omega |\omega\rangle$ is a quantum state in $\mathcal{H}_\Omega$, then measuring it in the basis $\{|\omega\rangle \mid \omega \in \Omega\}$ outputs index $\omega \in \Omega$ with probability $|\alpha_\omega|^2$.

We say that an algorithm is a randomized classical algorithm if the classical algorithm has access to random bits. In this paper, arithmetic (i.e., addition and multiplication) for real numbers are performed in $O(1)$ time.

In this paper, we use the total variation distance to measure the distance between two probability distributions.

Definition 4 (Total variation distance)**.** *The total variation distance $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}')$ between two distributions $\mathcal{D}$ and $\mathcal{D}'$ over a finite set $\mathcal{X}$ is defined as*

$$d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') = \frac{1}{2} \sum_{x \in \mathcal{X}} |\mathcal{D}(x) - \mathcal{D}'(x)|. \tag{15}$$

It is easy to see that the total variation distance is a metric, i.e., it satisfies the triangle inequality, and $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') = 0$ if and only if $\mathcal{D} = \mathcal{D}'$, and $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') \leq 1$ for any pair of $\mathcal{D}$ and $\mathcal{D}'$.

A.2 Quantum algorithmic subroutines

We recall two standard subroutines used in quantum algorithms for linear algebra, i.e., state preparation and block-encoding. These notions work as unitary transformations used in a quantum algorithm, rather than as black-box access to vectors or matrices.

Definition 5 (State-preparation unitary)**.** *Let $\boldsymbol{v} \in \mathbb{C}^n$ be a nonzero vector. A unitary $\boldsymbol{U}$ is called a state-preparation unitary for $\boldsymbol{v}$ if it prepares the normalized amplitude encoding of $\boldsymbol{v}$,*

$$\boldsymbol{U} |0\rangle = \frac{|\boldsymbol{v}\rangle}{\| |\boldsymbol{v}\rangle \|_2} \propto \sum_{i \in [n]} \frac{v_i}{\| |\boldsymbol{v}\rangle \|_2} |i\rangle , \tag{16}$$

where $|0\rangle$ denotes a fixed quantum state on a quantum register.

State preparation is known as a useful QML technique since a vector is encoded into amplitudes in a quantum state. In particular, for a probability distribution $\mathcal{D}$ over a finite set Ω, the quantum state

$$\left|\sqrt{\mathcal{D}}\right\rangle = \sum_{\omega \in \Omega} \sqrt{\mathcal{D}(\omega)} |\omega\rangle \tag{17}$$

is normalized, and its measurement in the computational basis outputs $\omega \in \Omega$ with probability $\mathcal{D}(\omega)$, i.e., state preparation gives a quantum sampling procedure for the probability distribution $\mathcal{D}$. Also, the state preparation is used for constructing a block-encoding of a matrix [15].

We introduce a definition of block-encoding [15]. Block-encoding is the standard way to represent a matrix inside a unitary operator.

Definition 6 (Block encoding)**.** *Let $\boldsymbol{A} \in \mathbb{C}^{t \times t}$, $\alpha > 0$, $a \in \mathbb{N}$, and $\epsilon > 0$. A unitary $\boldsymbol{U}$ acting on $a + \lceil \log t \rceil$ qubits is called an (α, a, ϵ)-block-encoding of $\boldsymbol{A}$ if*

$$\left\| \boldsymbol{A} - \alpha \left(\langle 0|^{\otimes a} \otimes \boldsymbol{I} \right) \boldsymbol{U} \left(|0\rangle^{\otimes a} \otimes \boldsymbol{I} \right) \right\|_{\mathrm{op}} \leq \epsilon. \tag{18}$$

Namely, if the unitary operator $\boldsymbol{U}$ is written in block form with respect to the auxiliary register, the block obtained by projecting the auxiliary register onto $|0\rangle^{\otimes a}$ is an approximation of $\boldsymbol{A}/\alpha$, where α is a normalization factor, a is the number of auxiliary qubits, and ϵ is an error parameter.

Block-encoding is a particularly crucial primitive for quantum singular value transformation (QSVT) since QSVT techniques apply polynomial transformations to the singular values of a block-encoded matrix. The quantum algorithm for sampling from the optimized probability distribution in [11] uses this technique to implement the inversion of the high-dimensional matrix $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}$.

A.3 Sampling and query access

We introduce the definition of sampling and query access oracles that are widely used in quantum-inspired classical algorithms [21, 15, 22, 31]. The motivation for these models comes from the input primitives used in QSVT algorithms. As discussed above, many QML algorithms begin by preparing a quantum state. Then, the algorithms apply unitary transformations such as block encodings to that quantum state. However, such state-preparation and block-encoding unitary operations are not efficiently implemented in general, and they may require some structure of the input data to be efficiently constructed.

A typical way to implement state preparation is to build a data structure that stores partial norms of a vector in a binary tree [30, 29]. This type of construction allows one to prepare a quantum state whose amplitudes are proportional to the entries of the encoded vector. On the other hand, the quantum-inspired algorithm proposed in [20] suggests that the same tree structure enables us to realize classical counterparts, i.e., querying an entry, sampling an index with a probability proportional to the squared magnitude of the entry, and computing the norm of the vector. The basic idea of quantum-inspired algorithms uses this structure to classically simulate the quantum operations.

Definition 7 (Query access [21])**.** *For a vector $\boldsymbol{v} \in \mathbb{C}^n$, let $\mathrm{Q}(\boldsymbol{v})$ be an oracle that, for any $i \in [n]$, takes as input $i \in [n]$ and outputs the value of v_i. Similarly, for a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, let $\mathrm{Q}(\boldsymbol{A})$ be an oracle that, for any $(i,j) \in [m] \times [n]$, takes as input (i,j) and outputs the value of $\boldsymbol{A}(i,j)$. Let $\mathbf{q}(\boldsymbol{v})$ and $\mathbf{q}(\boldsymbol{A})$ be the time complexities of one query to the oracles $\mathrm{Q}(\boldsymbol{v})$ and $\mathrm{Q}(\boldsymbol{A})$, respectively.*

Definition 8 (Sampling and query access to a vector [21])**.** *For a vector $\boldsymbol{v} \in \mathbb{C}^n$, let $\mathrm{SQ}(\boldsymbol{v})$ be an oracle that performs the following operations:*

- *For any $i \in [n]$, take as input i and output the value of v_i.*
- *Output an index $i \in [n]$ with probability $\frac{|v_i|^2}{\|\boldsymbol{v}\|_2^2}$.*
- *Output the value of $\|\boldsymbol{v}\|_2$.*

Let $\mathbf{q}(\boldsymbol{v}), \mathbf{s}(\boldsymbol{v})$, and $\mathbf{n}(\boldsymbol{v})$ be the time complexities of querying entries, sampling indices, and computing the norm, respectively. Let $\mathbf{sq}(\boldsymbol{v}) \coloneqq \max(\mathbf{q}(\boldsymbol{v}), \mathbf{s}(\boldsymbol{v}), \mathbf{n}(\boldsymbol{v}))$.

Block-encoding of a matrix is usually constructed from additional structures of the matrix, e.g., its sparsity and a binary tree description of the matrix. Thus, the efficient quantum implementation of a matrix, in general, depends on a data structure or oracle that provides sufficient information about the input matrix. Similar to accessing a vector, the sampling and query accesses to a matrix are defined using the sampling and query accesses to vectors.

Definition 9 (Sampling and query access to a matrix [21])**.** *For a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ and a vector $\boldsymbol{a} \in \mathbb{R}^m$ such that $a_i \coloneqq \|A_{i,*}\|_2$, let $\mathrm{SQ}(\boldsymbol{A})$ be an oracle that performs the following operations:*

- *For any $i \in [m]$, acts as $\mathrm{SQ}(A_{i,*})$,*
- *Acts as $\mathrm{SQ}(\boldsymbol{a})$,*

The time complexities of querying entries, sampling rows, and computing row norms are denoted by $\mathbf{q}(\boldsymbol{A}) \coloneqq \max(\mathbf{q}(A_{i,*}), \mathbf{q}(\boldsymbol{a}))$, $\mathbf{s}(\boldsymbol{A}) = \max(\mathbf{s}(A_{i,*}), \mathbf{s}(\boldsymbol{a}))$, and $\mathbf{n}(\boldsymbol{A}) = \mathbf{n}(\boldsymbol{a})$, respectively. Let $\mathbf{sq}(\boldsymbol{A}) \coloneqq \max(\mathbf{q}(\boldsymbol{A}), \mathbf{s}(\boldsymbol{A}), \mathbf{n}(\boldsymbol{A}))$.

However, constructing the block-encoding used in the quantum ridgelet sampling algorithm does not depend on the binary-tree data structure of the matrix $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}$, nor on the sparsity assumption. Therefore, existing sampling-based quantum-inspired techniques proposed in [21, 22, 20] cannot be applied directly. This motivates the need for a more problem-specific classical simulation developed in the proof of our main classical sampling theorem.

A.4 Discrete ridgelet transform and optimized probability distribution

We introduce the discrete ridgelet transform and its application to sparse subnetwork construction (see Yamasaki et al. [11] for more detail).

We first define the discrete Fourier transform, which can be implemented on a quantum computer since it is a unitary transformation. For a function $f : \mathbb{Z}_P^D \to \mathbb{R}$, define the D-dimensional discrete Fourier transform, for all $\boldsymbol{u} \in \mathbb{Z}_P^D$, by

$$\mathcal{F}_D[f](\boldsymbol{u}) \coloneqq P^{-D/2} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) \exp\left(-\frac{2\pi i}{P}\boldsymbol{u}^\top \boldsymbol{x}\right). \tag{19}$$

Similarly, for $g : \mathbb{Z}_P \to \mathbb{R}$, define the 1-dimensional discrete Fourier transform, for all $v \in \mathbb{Z}_P$, by

$$\mathcal{F}_1[g](v) \coloneqq P^{-1/2} \sum_{b} g(b) \exp\left(-\frac{2\pi i}{P} vb\right). \tag{20}$$

We also assume the admissibility condition for an activation function g and a ridgelet function $r : \mathbb{Z}_P \to \mathbb{R}$,

$$C_{g,r} \coloneqq \sum_v \mathcal{F}_1[g](v)\overline{\mathcal{F}_1[r](v)} \neq 0. \tag{21}$$

In particular, if we choose $r = g$, then $C_{g,g} = 1$.

The following two results are the main facts about this discrete transform.

Lemma 10 (Fourier slice theorem for discrete ridgelet transform [11])**.** *For any $f : \mathbb{Z}_P^D \to \mathbb{R}$, any $\boldsymbol{a} \in \mathbb{Z}_P^D$, and any $v \in \mathbb{Z}_P$, it holds that*

$$\mathcal{F}_1[\mathcal{R}[f](\boldsymbol{a}, \cdot)](v) = \mathcal{F}_D[f](v\boldsymbol{a} \bmod P)\overline{\mathcal{F}_1[r](v)}. \tag{22}$$

Lemma 11 (Exact representation of function as discretized neural network [11])**.** *For any function $f : \mathbb{Z}_P^D \to \mathbb{R}$ and any point $\boldsymbol{x} \in \mathbb{Z}_P^D$, it holds that*

$$f(\boldsymbol{x}) = C_{g,r}^{-1}\mathcal{S}[\mathcal{R}[f]](\boldsymbol{x}). \tag{23}$$

Lemma 11 gives us considering $\mathcal{R}[f](\boldsymbol{a}, b)$ as a coefficient assigned to the hidden node parameterized by $(\boldsymbol{a}, b)$. Since the representation is overcomplete, the coefficients are not unique in general. Nevertheless, this gives us explicit coefficients that reconstruct the function f exactly on the finite domain $\mathbb{Z}_P^D$.

The discrete ridgelet transform $\boldsymbol{R} : \mathbb{R}^{P^D} \to \mathbb{R}^{P^{D+1}}$ is an isometry, i.e.,

$$\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}_{P^D}. \tag{24}$$

Therefore, an efficient quantum implementation of $\boldsymbol{R}$ on the quantum system is given in Algorithm 1 of [11]. The quantum ridgelet transform is the quantum algorithm that implements this isometry efficiently by using $|r\rangle = \sum_b r(b)\,|b\rangle$, where $|r\rangle$ and $\mathrm{SQ}(r)$ can be prepared efficiently in runtime $O(\mathrm{polylog}(P))$. Given $|\psi\rangle \otimes |r\rangle$, the algorithm applies a D-dimensional quantum Fourier transform and a one-dimensional inverse quantum Fourier transform to the first and second register, respectively, performs a controlled map $|\boldsymbol{a}\rangle\,|v\rangle \mapsto \left|v^{-1}\boldsymbol{a}\right\rangle|v\rangle$, and finally applies the inverse quantum Fourier transform to the second register. By the Fourier slice theorem in Lemma 10, this construction implements

$$|\psi\rangle\,|r\rangle \mapsto \boldsymbol{R}\,|\psi\rangle \tag{25}$$

within runtime $O(D\,\mathrm{polylog}(P))$, where $|\psi\rangle$ is a quantum state on the input-domain Hilbert space $\mathbb{C}^{P^D}$.

We summarize how the optimized probability distribution in Definition 1 used for sparse-subnetwork construction is defined. For readability, we briefly restate the setting of sampling. As mentioned in the manuscript, we take $r = g$. Let $\{\boldsymbol{x}_m, y_m\}_{m\in[M]}$ be a training dataset of M pairs of samples with $y_m = f(\boldsymbol{x}_m)$. Let $\hat{p}_{\mathrm{data}}$ be the empirical distribution of the input data and define

$$\hat{\boldsymbol{P}}_{\mathrm{data}} \coloneqq \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})\,|\boldsymbol{x}\rangle\langle\boldsymbol{x}|\,. \tag{26}$$

We also write

$$|\psi_{\mathrm{in}}\rangle \coloneqq \hat{\boldsymbol{P}}_{\mathrm{data}}\,|f\rangle = \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})f(\boldsymbol{x})\,|\boldsymbol{x}\rangle\,, \tag{27}$$

where this vector is not necessarily normalized. Consider the regularized weighted least-squares regression to be minimized in (2)

$$\tilde{J}(w) = \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})|f(\boldsymbol{x}) - \mathcal{S}[w](\boldsymbol{x})|^2 + \lambda \sum_{\boldsymbol{a},b} \left|P^{-D/2}w(\boldsymbol{a}, b)\right|^2. \tag{28}$$

This can be written by using the matrix notation as

$$\tilde{J}(w) = \|\hat{\boldsymbol{P}}_{\mathrm{data}}^{1/2}(|f\rangle - \boldsymbol{R}^\top\,|w\rangle)\|_2^2 + \lambda P^{-D}\|\,|w\rangle\,\|_2^2, \tag{29}$$

where $|w\rangle = \sum_{\boldsymbol{a},b} w(\boldsymbol{a}, b)\,|\boldsymbol{a}, b\rangle$. The regularization term makes the minimization of $\tilde{J}(w)$ strongly convex in $|w\rangle$. By differentiating this quadratic function in $|w\rangle$, we obtain the following equation

$$\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)|w_\lambda^*\rangle = \boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\,|f\rangle = \boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle\,. \tag{30}$$

Therefore, the optimal solution vector is given by

$$|w_\lambda^*\rangle = \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle\,. \tag{31}$$

Namely, the coefficient on the hidden node parameterized by $(\boldsymbol{a}, b)$ is $w_\lambda^*(\boldsymbol{a}, b) = \langle \boldsymbol{a}, b|w_\lambda^*\rangle$.

In the manuscript, the optimized probability distribution $p_{\lambda,\Delta}^*$ is defined as

$$p_{\lambda,\Delta}^*(\boldsymbol{a}, b) \coloneqq \frac{1}{Z}\frac{|P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)|^2}{|P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)|^2 + \Delta}, \qquad \Delta > 0, \tag{32}$$

where

$$Z \coloneqq \sum_{\boldsymbol{a},b}\frac{|P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)|^2}{|P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)|^2 + \Delta} \tag{33}$$

is the normalization constant, and the parameter Δ smooths the distribution. The corresponding quantum sampling state is

$$\left|p_{\lambda,\Delta}^*\right\rangle = \frac{1}{\sqrt{Z}}\sum_{\boldsymbol{a},b}\frac{P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)}{\sqrt{|P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)|^2 + \Delta}}\,|\boldsymbol{a}, b\rangle\,. \tag{34}$$

To represent this in matrix form, we define a diagonal matrix $\boldsymbol{W}_\lambda$ by

$$\boldsymbol{W}_\lambda \coloneqq \frac{1}{\gamma}\sum_{\boldsymbol{a},b}|P^{-D/2}w_\lambda^*(\boldsymbol{a}, b)|^2\,|\boldsymbol{a}, b\rangle\langle\boldsymbol{a}, b|\,, \tag{35}$$

where γ is defined in (3). Then, we can write a vector proportional to $\left|p_{\lambda,\Delta}^*\right\rangle$ as

$$\left|p_{\lambda,\Delta}^*\right\rangle \propto \left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-1/2}\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle\,. \tag{36}$$

Measuring this state in the computational basis outputs $(\boldsymbol{a}, b)$ according to $p_{\lambda,\Delta}^*$. Thus, the quantum algorithm constructs a sparse subnetwork by repeatedly preparing this state and measuring it. The quantum ridgelet-transform algorithm prepares this state by combining state preparation, the quantum ridgelet transform, block encodings, and QSVT [11].

B Key technical lemma

In this section, we prove the identity that is used throughout the proof of both our classical and improved quantum sampling theorems. The computational difficulty in sampling from the optimized probability distribution $p_{\lambda,\Delta}^*$ is the inverse

$$\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1}. \tag{37}$$

This acts on the parameter space of dimension P^{D+1}, and the matrix is not sparse in the standard basis. The following lemma states a special case of the Woodbury identity [26, 27] by using the property of $\boldsymbol{R}$, i.e., $\boldsymbol{R}$ is an isometry.

Lemma 12 (Inverse decomposition)**.**

$$\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1} = \boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}^\top + \frac{1}{\lambda P^{-D}}(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top) \tag{38}$$

Proof. We can check the equality by applying the right-hand side to the original matrix, i.e.,

$$\begin{aligned}
&\left\{\boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}^\top + \frac{1}{\lambda P^{-D}}(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top)\right\}\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)\\
&= \boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}^\top + (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top)\\
&= \boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}^\top + (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top)\\
&= \boldsymbol{I}.
\end{aligned} \tag{39}$$

Algorithm 1: Classical sampling from the optimized probability distribution

Input: precision parameters $\lambda, \Delta > 0$, $\delta \in (0,1)$, and γ in (3), and the data set $\{\boldsymbol{x}_m, y_m\}_{m\in[M]}$.

Output: A random variable $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ sampled from a probability distribution $\mathcal{D}'$ close to the distribution $p^*_{\lambda,\Delta}$ in Definition 1 at most δ, i.e., $d_{\mathrm{TV}}(\mathcal{D}', p^*_{\lambda,\Delta}) \leq \delta$.

Let K be the number of distinct $\boldsymbol{x}_m$'s, i.e. $K = |\{\boldsymbol{x}_m \mid m \in [M]\}|$;

Set

$$I = \left\lceil K\left(1 + \frac{\gamma}{\Delta}\right)\ln(1/\delta)\right\rceil \tag{42}$$

and $A = 0$;

Construct a data structure to implement $\mathrm{SQ}(|\phi\rangle)$ to $|\phi\rangle$ in (47);

for $i \in [I]$ **do**

  Sample $\boldsymbol{x} \in \mathbb{Z}_P^D$ from the distribution proportional to $|\phi\rangle$ by $\mathrm{SQ}(|\phi\rangle)$;

  Sample $\boldsymbol{a} \in \mathbb{Z}_P^D$ from the uniform distribution over $\mathbb{Z}_P^D$;

  Sample $t \in \mathbb{Z}_P$ with probability $|r(t)|^2$ with $\mathrm{SQ}(r)$, and let $b = \boldsymbol{a}^\top \boldsymbol{x} - t$;

  Compute $\frac{\Delta}{\Delta + s(\boldsymbol{a},b)} \frac{s(\boldsymbol{a},b)}{K\gamma q(\boldsymbol{a},b)}$ in (51); Sample u from the uniform distribution over $[0,1]$;

  if $u \leq \frac{\Delta}{\Delta + s(\boldsymbol{a},b)} \frac{s(\boldsymbol{a},b)}{K\gamma q(\boldsymbol{a},b)}$ **then**

    Set $A = 1$;

  end

  if $A = 1$ **then**

    break;

  end

end

if $A = 0$ **then**

  Sample $(\boldsymbol{a}, b)$ uniformly from $\mathbb{Z}_P^D \times \mathbb{Z}_P$;

end

return $(\boldsymbol{a}, b)$;

Since $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}$ is positive definite, due to $\lambda > 0$, its inverse exists, which yields the conclusion. □

The key consequence from this lemma is that the second term in (38) vanishes when the inverse is applied to a vector in the image of $\boldsymbol{R}$, i.e.,

$$(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\top)\boldsymbol{R} = 0. \tag{40}$$

Therefore, the optimal solution $|w^*_\lambda\rangle$ in (31) is reduced to

$$\begin{aligned} |w^*_\lambda\rangle &= \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\boldsymbol{R}\,|\psi_{\mathrm{in}}\rangle \\ &= \boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}|\psi_{\mathrm{in}}\rangle\,. \end{aligned} \tag{41}$$

C Proof of Theorem 2

In this Appendix, we give the proof of Theorem 2 by using Algorithm 1 with Lemma 12.

Theorem 13 (The restatement of Theorem 2)**.** *Given parameters $\lambda, \Delta > 0$, $\delta \in (0,1)$, and γ in* (3)*, and the dataset $\{\boldsymbol{x}_m, y_m\}_{m\in[M]}$, Algorithm 1 outputs one sample from a distribution $\mathcal{D}'$ whose total variation distance from $p^*_{\lambda,\Delta}$ in Definition 1 is at most δ, i.e.,*

$$d_{\mathrm{TV}}(p^*_{\lambda,\Delta}, \mathcal{D}') \leq \delta, \tag{43}$$

with the sample complexity

$$\tilde{O}\left(DM^2\left(1 + \frac{\gamma}{\Delta}\right)\log\left(\frac{1}{\delta}\right)\mathrm{polylog}(P)\right). \tag{44}$$

Proof. We first show the correctness of Algorithm 1, and then we analyze the runtime.

Let $\mathcal{X}_{\mathrm{data}}$ be the support of the empirical distribution $\hat{p}_{\mathrm{data}}$, i.e. $\mathcal{X}_{\mathrm{data}} = \{x_m \mid m \in [M]\}$ and let $K \coloneqq |\mathcal{X}_{\mathrm{data}}| \leq M$. For simplicity of notation, let us write $s(\boldsymbol{a}, b) = |P^{-D/2} w^*_\lambda(\boldsymbol{a}, b)|^2$, i.e.,

$$\begin{aligned} s(\boldsymbol{a}, b) &= \left| \langle \boldsymbol{a}, b | \, P^{-D/2} \boldsymbol{R} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}} \, |f\rangle \right|^2, \\ &= \left| \sum_{\boldsymbol{x}} P^{-D} r\left((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P \right) \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \right|^2, \end{aligned} \tag{45}$$

from (41), and thus it holds that

$$\begin{aligned} &\sum_{\boldsymbol{a}, b} s(\boldsymbol{a}, b) \\ =& P^{-D} \sum_{\boldsymbol{a}, b} \langle f | \, \hat{\boldsymbol{P}}_{\mathrm{data}} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \boldsymbol{R}^\top \, |\boldsymbol{a}, b\rangle \, \langle \boldsymbol{a}, b | \, \boldsymbol{R} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}} \, |f\rangle \\ =& P^{-D} \, \langle f | \, \hat{\boldsymbol{P}}_{\mathrm{data}} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \boldsymbol{R}^\top \boldsymbol{I} \boldsymbol{R} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}} \, |f\rangle \\ =& P^{-D} \, \langle f | \, \hat{\boldsymbol{P}}_{\mathrm{data}} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}} \, |f\rangle \\ =& \| P^{-D/2} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}} \, |f\rangle \, \|_2^2. \end{aligned} \tag{46}$$

Algorithm 1 tries to sample from the distribution $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ in Definition 1 by rejection sampling. To achieve this, we construct a data structure to implement sampling and query access oracle $\mathrm{SQ}(|\phi\rangle)$ to the vector

$$\begin{aligned} |\phi\rangle &\coloneqq P^{-D/2} (\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D} \boldsymbol{I})^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}} \, |f\rangle \\ &= P^{-D/2} \sum_{\boldsymbol{x}} \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \, |\boldsymbol{x}\rangle \, , \end{aligned} \tag{47}$$

which can be achieved within $\tilde{O}(M)$ time by constructing the binary-tree structure used in [21, 30, 11], where the query complexity

$$\mathbf{sq}(|\phi\rangle) = \mathrm{polylog}(M), \tag{48}$$

and $\| \, |\phi\rangle \, \|_2^2 = \gamma$ from (3).

Then, we define the proposal probability distribution q over $\mathbb{Z}_P^D \times \mathbb{Z}_P$ for the rejection sampling from $p^*_{\lambda,\Delta}$ as follows,

$$q(\boldsymbol{a}, b) \coloneqq \frac{1}{\gamma} \sum_{\boldsymbol{x}} \left| P^{-D/2} \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \right|^2 \left| P^{-D/2} r\left((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P \right) \right|^2. \tag{49}$$

Algorithm 1 samples from this distribution q by sampling $\boldsymbol{x} \in \mathbb{Z}_P^D$ from $|\phi\rangle$ with $\mathrm{SQ}(|\phi\rangle)$, $\boldsymbol{a} \in \mathbb{Z}_P^D$ uniformly, and $t \in \mathbb{Z}_P$ with probability $|r(t)|^2$ with $\mathrm{SQ}(r)$, and letting $b = \boldsymbol{a}^\top \boldsymbol{x} - t$. The probability of obtaining $(\boldsymbol{a}, b)$ in this process holds

$$\begin{aligned} \Pr(\boldsymbol{a}, b) &= \sum_{\boldsymbol{x}} \Pr(\boldsymbol{x}, \boldsymbol{a}, b) \\ &= \sum_{\boldsymbol{x}} \Pr(\boldsymbol{x}, \boldsymbol{a}, t) \\ &= \sum_{\boldsymbol{x}} \Pr(\boldsymbol{x}) \Pr(\boldsymbol{a}) \Pr(t) \\ &= \sum_{\boldsymbol{x}} \frac{\left| P^{-D/2} \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \right|^2}{\| \, |\phi\rangle \, \|_2^2} \frac{1}{P^D} |r(t)|^2 \\ &= \sum_{\boldsymbol{x}} \frac{\left| P^{-D/2} \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \right|^2}{\| \, |\phi\rangle \, \|_2^2} \frac{1}{P^D} |r(\boldsymbol{a}^\top \boldsymbol{x} - b)|^2 \\ &= q(\boldsymbol{a}, b). \end{aligned} \tag{50}$$

Therefore, Algorithm 1 exactly samples from q.

Then, we accept $(\boldsymbol{a}, b)$ sampled from the probability distribution q with probability

$$\frac{\Delta}{\Delta + s(\boldsymbol{a}, b)} \frac{s(\boldsymbol{a}, b)}{K\gamma q(\boldsymbol{a}, b)} \leq 1, \tag{51}$$

where the inequality holds from the Cauchy-Schwarz inequality. The probability $\Pr(\boldsymbol{a}, b : \text{accept})$ of each $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ is returned is given by

$$\begin{aligned} \Pr(\boldsymbol{a}, b : \text{accept}) &= \frac{\Delta}{\Delta + s(\boldsymbol{a}, b)} \frac{s(\boldsymbol{a}, b)}{K\gamma q(\boldsymbol{a}, b)} q(\boldsymbol{a}, b) \\ &= \frac{\Delta}{K\gamma} \frac{s(\boldsymbol{a}, b)}{\Delta + s(\boldsymbol{a}, b)}. \end{aligned} \tag{52}$$

Thus, the probability that a sample drawn from q is accepted in Algorithm 1 is

$$\begin{aligned} \Pr(\text{accept}) &= \sum_{\boldsymbol{a}, b} \Pr(\boldsymbol{a}, b : \text{accept}) \\ &= \frac{\Delta}{K\gamma} \sum_{\boldsymbol{a}, b} \frac{s(\boldsymbol{a}, b)}{\Delta + s(\boldsymbol{a}, b)}, \end{aligned} \tag{53}$$

Notice that the lower bound of $\Pr(\text{accept})$ is obtained as follows

$$\begin{aligned} \Pr(\text{accept}) &= \frac{\Delta}{K\gamma} \sum_{\boldsymbol{a}, b} \frac{s(\boldsymbol{a}, b)}{\Delta + s(\boldsymbol{a}, b)} \\ &\geq \frac{\Delta}{K\gamma} \sum_{\boldsymbol{a}, b} \frac{s(\boldsymbol{a}, b)}{\Delta + \gamma} \\ &= \frac{\Delta}{K\gamma} \frac{\gamma}{\Delta + \gamma} \\ &= \frac{1}{K} \frac{\Delta}{\Delta + \gamma}, \end{aligned} \tag{54}$$

where we use the fact that $s(\boldsymbol{a}, b) \leq \gamma$ in the inequality. Therefore, the probability distribution returned by Algorithm 1 when the rejection sampling succeeds is given by

$$\begin{aligned} \Pr(\boldsymbol{a}, b) &= \frac{\Pr(\boldsymbol{a}, b : \text{accept})}{\Pr(\text{accept})} \\ &= \frac{s(\boldsymbol{a}, b)}{\Delta + s(\boldsymbol{a}, b)} \left(\sum_{\boldsymbol{a}, b} \frac{s(\boldsymbol{a}, b)}{\Delta + s(\boldsymbol{a}, b)} \right)^{-1} \\ &= p^*_{\lambda, \Delta}(\boldsymbol{a}, b). \end{aligned} \tag{55}$$

Algorithm 1 repeats this sampling for I times, and thus, the probability that the rejection sampling fails is given by

$$\begin{aligned} \Pr(\text{failure}) &= (1 - \Pr(\text{accept}))^I \\ &\leq \exp(-I \times \Pr(\text{accept})) \\ &\leq \exp\left(-I \times \frac{1}{K} \frac{\Delta}{\Delta + \gamma} \right) \\ &= \exp\left(-\left\lceil K \left(1 + \frac{\gamma}{\Delta} \right) \ln(1/\delta) \right\rceil \times \frac{1}{K} \frac{\Delta}{\Delta + \gamma} \right) \\ &\leq \exp\left(-K \left(1 + \frac{\gamma}{\Delta} \right) \ln(1/\delta) \times \frac{1}{K} \frac{\Delta}{\Delta + \gamma} \right) \\ &\leq \delta. \end{aligned} \tag{56}$$

Algorithm 2: Quantum sampling from the optimized distribution

Input: precision parameters $\lambda, \Delta, \delta > 0$ and γ in (3), and the data set $\{\boldsymbol{x}_m, y_m\}_{m\in[M]}$.

Output: A random variable $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ sampled from a probability distribution $\mathcal{D}'$ close to the distribution $p^*_{\lambda,\Delta}$ in Definition 1 at most δ, i.e., $d_{\mathrm{TV}}(\mathcal{D}', p^*_{\lambda,\Delta}) \leq \delta$.

Prepare a quantum state $\left|\hat{\phi}\right\rangle \propto \sum_{\boldsymbol{x}} \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} |\boldsymbol{x}\rangle$;

Apply quantum ridgelet transform to obtain a state $\boldsymbol{R}\left|\hat{\phi}\right\rangle$;

Apply $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I})^{-1/2}$ by QSVT to obtain

$$\left|p^*_{\lambda,\Delta}\right\rangle \propto \left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-1/2} \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1} \boldsymbol{R}\left|\psi_{\mathrm{in}}\right\rangle. \tag{59}$$

Perform a quantum measurement in the standard basis to sample $(\boldsymbol{a}, b)$ with the probability $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$.;

return $(\boldsymbol{a}, b)$;

When the sampling fails, Algorithm 1 outputs $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ sampled uniformly. Since the output distribution conditioned on successful rejection sampling is exactly $p^*_{\lambda,\Delta}$, the total variation distance between the output distribution $\mathcal{D}'$ of Algorithm 1 and $p^*_{\lambda,\Delta}$ satisfies

$$d_{\mathrm{TV}}(\mathcal{D}', p^*_{\lambda,\Delta}) \leq \Pr(\text{failure}) \leq \delta. \tag{57}$$

This completes the proof of correctness in Algorithm 1.

Finally, we analyze the runtime of Algorithm 1. The runtime of pre-processing is dominated by constructing the data structure of $\mathrm{SQ}(|\phi\rangle)$, which takes $\tilde{O}(M)$. The runtime of Algorithm 1 is longest when the algorithm repeats I times, and that fails. In this case, the runtime is dominated by the runtime of I computations of the acceptance rate in Equation (51). The dominant part of computing this value is to compute $s(\boldsymbol{a}, b)$ in Equation (45). To compute this value, we compute inner products $\boldsymbol{a}^\top \boldsymbol{x}$, query $\mathrm{SQ}(r)$ and $\mathrm{SQ}(|\phi\rangle)$ K times and sum of the K values, i.e., it takes $O(K(D + \mathbf{sq}(r) + \mathbf{sq}(|\phi\rangle)))$. Therefore, the total runtime of Algorithm 1 is upper bounded by

$$I \times K(D + \mathbf{sq}(r) + \mathbf{sq}(|\phi\rangle)) = O(D\log(1/\delta)\mathrm{polylog}(P)) \times \tilde{O}\left(M^2\left(1 + \frac{\gamma}{\Delta}\right)\right), \tag{58}$$

which yields the conclusion of Theorem 13. □

D Proof of Theorem 3

Theorem 14 (The restatement of Theorem 3)**.** *Given parameters $\lambda, \Delta, \delta > 0$, γ in* (3)*, and the dataset $\{(\boldsymbol{x}_m, y_m)\}_{m\in[M]}$, Algorithm 2 outputs one sample from a distribution $\mathcal{D}'$ satisfying*

$$d_{\mathrm{TV}}(\mathcal{D}', p^*_{\lambda,\Delta}) \leq \delta \tag{60}$$

with the sample complexity

$$\widetilde{O}\left(D\left(1 + \frac{\gamma}{\Delta}\right)\mathrm{polylog}\frac{MP}{\delta}\right). \tag{61}$$

after preprocessing $\tilde{O}(M)$.

Proof. Algorithm 2 basically follows the quantum algorithm presented in [11], but uses Lemma 12 to avoid implementing the inverse

$$\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right)^{-1} \tag{62}$$

by QSVT. We first analyze the correctness of Algorithm 2, and then summarize its total runtime.

By Lemma 12, we have the following representation of the optimal solution in (2)

$$|w^*_\lambda\rangle = \boldsymbol{R}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1} \hat{\boldsymbol{P}}_{\mathrm{data}}|f\rangle. \tag{63}$$

Similar to Algorithm 1, we define a vector

$$|\phi\rangle = P^{-D/2}\left(\hat{\boldsymbol{P}}_{\mathrm{data}} + \lambda P^{-D}\boldsymbol{I}\right)^{-1}\hat{\boldsymbol{P}}_{\mathrm{data}}\,|f\rangle\,, \tag{64}$$

and since $\hat{\boldsymbol{P}}_{\mathrm{data}}$ is diagonal, we have the explicit form

$$|\phi\rangle = P^{-D/2}\sum_{\boldsymbol{x}}\frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x})f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})}\,|\boldsymbol{x}\rangle\,. \tag{65}$$

In particular, the support of $\hat{\boldsymbol{P}}_{\mathrm{data}}$ is at most $O(M)$ and we can evaluate each value in constant time. Therefore, as in the original quantum sampling in [11], Algorithm 2 prepares the quantum state proportional to $|\phi\rangle$ within $\tilde{O}(\mathrm{polylog}(M))$ by constructing the binary-tree structure [29, 30] which takes $O(M)$. Namely, we can construct a unitary operation $\boldsymbol{U}$ such that

$$\boldsymbol{U}\,|0\rangle = \frac{|\phi\rangle}{\|\,|\phi\rangle\,\|_2} =: \left|\hat{\phi}\right\rangle \tag{66}$$

implemented by a $\mathrm{polylog}(M)$-depth quantum circuit after $\tilde{O}(M)$ preprocessing.

By (63) and (64), we have

$$|w_\lambda^*\rangle = P^{D/2}\boldsymbol{R}\,|\phi\rangle\,. \tag{67}$$

Thus, after preparing $\left|\hat{\phi}\right\rangle$ and applying the quantum ridgelet transform, Algorithm 2 obtains

$$\boldsymbol{R}\left|\hat{\phi}\right\rangle = \frac{1}{\sqrt{\gamma}}\sum_{\boldsymbol{a},b}P^{-D/2}w_\lambda^*(\boldsymbol{a},b)\,|\boldsymbol{a},b\rangle\,, \tag{68}$$

which takes $O(D\ \mathrm{polylog}(MP))$ runtime.

We construct the block-encoding of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$, where $\boldsymbol{W}_\lambda$ in (35). This can be done by using the block-encoding unitaries of $\boldsymbol{W}_\lambda$ and $\boldsymbol{I}$. The block-encoding of I is a trivial unitary operator, and thus, consider the block-encoding of $\boldsymbol{W}_\lambda$. The unitary operator $\boldsymbol{U}$ in (66) and the quantum ridgelet transform $\boldsymbol{R}$ produce the purification of $\boldsymbol{W}_\lambda$. Namely, adding an auxiliary register and applying CNOT to $\boldsymbol{R}\left|\hat{\phi}\right\rangle$, we obtain a quantum state

$$\frac{1}{\sqrt{\gamma}}\sum_{\boldsymbol{a},b}P^{-D/2}w_\lambda^*(\boldsymbol{a},b)\,|\boldsymbol{a},b\rangle\otimes|\boldsymbol{a},b\rangle\,. \tag{69}$$

By tracing out the second register, this exactly yields

$$\boldsymbol{W}_\lambda \coloneqq \frac{1}{\gamma}\sum_{\boldsymbol{a},b}|P^{-D/2}w_\lambda^*(\boldsymbol{a},b)|^2\,|\boldsymbol{a},b\rangle\langle\boldsymbol{a},b|\,, \tag{70}$$

then we can construct a $(1, O((D+1)\log P), \delta)$-block encoding of $\boldsymbol{W}_\lambda$ by using $\boldsymbol{R}, \boldsymbol{R}^\dagger, \boldsymbol{U}$, and $\boldsymbol{U}^\dagger$ a constant number of times [15, 11]. The construction of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$ can be implemented by using the block-encoding of $\boldsymbol{W}_\lambda$ and $\boldsymbol{I}$ a constant number of times.

To apply $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I})^{-1/2}$, we use QSVT of the block-encoding of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$, which can be implemented by using the block-encoding and its conjugate transpose linear times in the condition number of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$ up to logarithmic factor, i.e. $\tilde{O}(1+\gamma/\Delta)\times\mathrm{polylog}(1/\delta)$ times [32, 33, 15], where δ is chosen appropriately for the desired accuracy. In summary, Algorithm 2 prepares a quantum state proportional to (64), applies the quantum ridgelet transform $\boldsymbol{R}$ to the state, and then, applies $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I})^{-1/2}$, which yields a quantum state $\left|p_{\lambda,\Delta}^*\right\rangle$ in (36). This completes the proof of correctness of Algorithm 2.

The runtime of Algorithm 2 is dominated by applying $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I})^{-1/2}$. To implement this, we require $\tilde{O}(1+\gamma/\Delta)\times\mathrm{polylog}(1/\delta)$ preparations of $\boldsymbol{R}\left|\hat{\phi}\right\rangle$. Therefore, the total runtime is

$$O(D\ \mathrm{polylog}(MP))\times\tilde{O}\left(1+\frac{\gamma}{\Delta}\right)\times\mathrm{polylog}\left(\frac{1}{\delta}\right) = O\left(D\ \mathrm{polylog}\left(\frac{MP}{\delta}\right)\right)\times\tilde{O}\left(1+\frac{\gamma}{\Delta}\right). \tag{71}$$

□

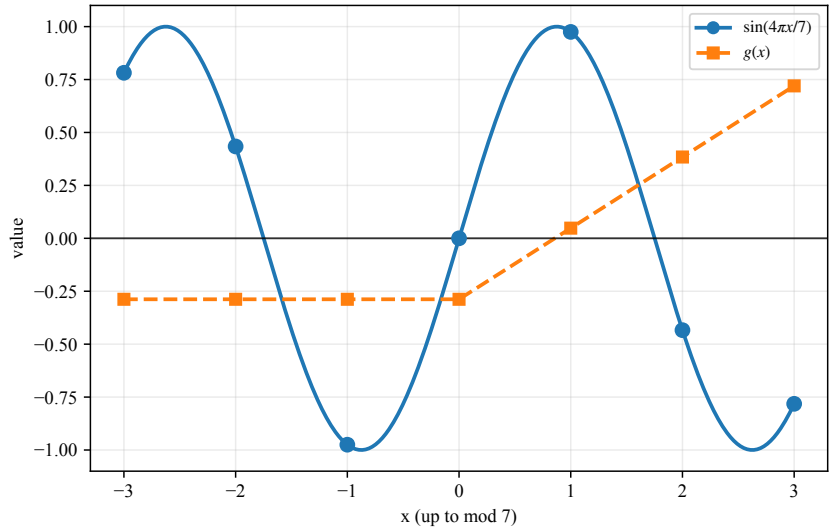

Figure 3: The function f_D to be learned and the activation function g, when the dimension $D = 1$ and $P = 7$.

E Details of Numerical Experiments

We describe the detailed setting used in numerical experiments in Figures 1 and 2. All experiments are performed in the finite-domain setting over $\mathbb{Z}_P$.

E.1 Empirical-risk experiment

For the empirical-risk experiment, we set

$$P = 7, \quad D \in \{1, 2, 3, 4, 5, 6\}, \quad M = 50D. \tag{72}$$

For each dimension, we generate the training inputs independently from the uniform distribution over $\mathbb{Z}_P^D$, i.e.,

$$p_{\mathrm{data}}(\boldsymbol{x}) = P^{-D}. \tag{73}$$

The empirical distribution $\hat{p}_{\mathrm{data}}$ is constructed from these training data. The labels are noiseless and are generated as $y_m = f_D(\boldsymbol{x}_m)$, where

$$f_D(\boldsymbol{x}) = \sin\left(\frac{4\pi}{P}\left(\sum_{d=1}^{D} x_d \bmod P\right)\right). \tag{74}$$

We use a centered and normalized discrete ReLU activation function, i.e.,

$$g(x) = r(x) \propto \begin{cases} (x \bmod P) - \frac{P^2-1}{8P}, & 0 \le (x \bmod P) \le (P-1)/2, \\ -\frac{P^2-1}{8P}, & (P+1)/2 \le (x \bmod P) \le P-1. \end{cases} \tag{75}$$

Figure 3 shows the function f_D and the activation function g for the case $D = 1, P = 7$.

The regularization parameter is set to

$$\lambda P^{-D} = 10^{-3}, \tag{76}$$

and we set

$$\Delta = \gamma. \tag{77}$$

We compare the following four sampling procedures:

- exact sampling from the optimized distribution $p^*_{\lambda,\Delta}$,

- our dequantized sampler with accuracy parameter $\delta = 0.1$,
- our dequantized sampler with accuracy parameter $\delta = 0.7$,
- uniform sampling over $\mathbb{Z}_P^D \times \mathbb{Z}_P$.

For the exact optimized sampler, we explicitly enumerate all $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ and compute the unnormalized weight

$$s(\boldsymbol{a}, b) = \left| \sum_{\boldsymbol{x}} P^{-D} r\left((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P\right) \frac{\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})}{\lambda P^{-D} + \hat{p}_{\mathrm{data}}(\boldsymbol{x})} \right|^2, \tag{78}$$

then compute the optimized distribution

$$p^*_{\lambda,\Delta}(\boldsymbol{a}, b) \coloneqq \frac{1}{Z} \frac{s(\boldsymbol{a}, b)}{s(\boldsymbol{a}, b) + \Delta}. \tag{79}$$

For each sampling procedure, we sample

$$N \in \{8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\} \tag{80}$$

nodes including overlaps.

For each sampling procedure, we first obtain a list of N sampled hidden nodes

$$T_N = ((\boldsymbol{a}_1, b_1), \ldots, (\boldsymbol{a}_N, b_N)) \in \left(\mathbb{Z}_P^D \times \mathbb{Z}_P\right)^N. \tag{81}$$

Since the sampling procedures output nodes with replacement, the same hidden node may appear multiple times in T_N. Before fitting the output weights, we remove duplicated hidden nodes and define

$$S_N \coloneqq \mathrm{unique}(T_N) = \left\{(\tilde{\boldsymbol{a}}_1, \tilde{b}_1), \ldots, (\tilde{\boldsymbol{a}}_{N_{\mathrm{eff}}}, \tilde{b}_{N_{\mathrm{eff}}})\right\}, \tag{82}$$

where

$$N_{\mathrm{eff}} \coloneqq |S_N| \le \min\{N, P^{D+1}\}. \tag{83}$$

Thus, N is the number of sampled proposals, whereas N_{eff} is the actual number of distinct hidden nodes used in the fitted sparse network.

Let

$$\mathrm{Supp}(\hat{p}_{\mathrm{data}}) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \tag{84}$$

be the empirical support. We write

$$\hat{p}_i \coloneqq \hat{p}_{\mathrm{data}}(\boldsymbol{x}_i), \qquad f_i \coloneqq f_D(\boldsymbol{x}_i), \qquad \boldsymbol{f}_{\mathrm{supp}} \coloneqq (f_1, \ldots, f_K)^\top, \tag{85}$$

and

$$\hat{\boldsymbol{P}}_{\mathrm{supp}} \coloneqq \mathrm{diag}(\hat{p}_1, \ldots, \hat{p}_K). \tag{86}$$

For the deduplicated sampled set S_N, we define the sampled design matrix

$$\Phi_{S_N} \in \mathbb{R}^{K \times N_{\mathrm{eff}}} \tag{87}$$

by

$$(\Phi_{S_N})_{ij} \coloneqq P^{-D/2} g\left((\tilde{\boldsymbol{a}}_j^\top \boldsymbol{x}_i - \tilde{b}_j) \bmod P\right), \qquad i \in [K],\ j \in [N_{\mathrm{eff}}]. \tag{88}$$

The output weights are obtained by ridge regression on this sampled design matrix:

$$\hat{\theta}_{S_N} \in \operatorname*{arg\,min}_{\theta \in \mathbb{R}^{N_{\mathrm{eff}}}} \left\| \hat{\boldsymbol{P}}_{\mathrm{supp}}^{1/2} \left(\Phi_{S_N} \theta - \boldsymbol{f}_{\mathrm{supp}}\right) \right\|_2^2 + \lambda P^{-D} \|\theta\|_2^2. \tag{89}$$

The reported finite empirical risk is

$$\widehat{\mathcal{R}}(S_N) \coloneqq \left\| \hat{\boldsymbol{P}}_{\mathrm{supp}}^{1/2} \left(\Phi_{S_N} \hat{\theta}_{S_N} - \boldsymbol{f}_{\mathrm{supp}}\right) \right\|_2^2 = \sum_{i=1}^{K} \hat{p}_{\mathrm{data}}(\boldsymbol{x}_i) \left((\Phi_{S_N} \hat{\theta}_{S_N})_i - f_D(\boldsymbol{x}_i)\right)^2. \tag{90}$$

We solve the corresponding normal equations using linear algebra routines in NumPy.

Deduplication is important in low-dimensional experiments. If identical hidden nodes were kept as separate columns of the design matrix, the same ridge function could be assigned multiple independent output weights, i.e., weights on the nodes may be non-unique. Deduplicating the sampled nodes ensures that the fitted model has one output weight per distinct hidden node.

In Figure 1, the horizontal axis reports the number of sampled proposals N. The actual number of fitted hidden nodes is N_{eff}, which is at most P^{D+1}. Therefore, when $N > P^{D+1}$ in low-dimensional settings, the fitted network saturates at no more than P^{D+1} distinct hidden nodes.

E.2 Runtime experiment

For the runtime experiment, we use the same data-generation procedure, target function, activation function, and regularization convention as in the empirical-risk experiment, except that we set $P = 3$. This choice is made only to keep the naive baseline executable. The hidden-node parameter space has size

$$|\mathbb{Z}_P^D \times \mathbb{Z}_P| = P^{D+1}. \tag{91}$$

Therefore, explicitly constructing matrices over the hidden-node space becomes rapidly infeasible as D grows. For example, for $P = 7$ and $D = 7$, the hidden-node space already has size

$$7^8 = 5,764,801, \tag{92}$$

whereas for $P = 3$ and $D = 8$, it has size

$$3^9 = 19,683. \tag{93}$$

The naive sampler explicitly constructs the matrix

$$\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}, \tag{94}$$

and solves the dense linear system

$$\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda P^{-D}\boldsymbol{I}\right) w = \boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{f}. \tag{95}$$

After computing w, it computes and samples the corresponding sampling weights over all $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$.

We run this naive sampler for

$$D \in \{1, 2, 3, 4, 5, 6, 7, 8\}, \tag{96}$$

or up to the largest dimension for which the dense linear solve is executable within memory and time limits.

We also run the classical sampler proposed in Algorithm 1. We measure the time required to output one sample. This proposed sampler is run for

$$D \in \{1, 2, 4, 8, 16, 32, 64, 128\}. \tag{97}$$

For each D and each method, we repeat the runtime measurement over 20 independent executions. Each point in Figure 2 shows the mean, and each error bar shows the 95% confidence interval of the mean.

All numerical experiments were conducted on a MacBook Pro with an Apple M4 Pro CPU and 48 GB unified memory. The experiments were run on CPU using Python and NumPy, and the reported runtimes are wall-clock times measured by `time.perf_counter()`.

F Limitations and Broader Impact

F.1 Limitations

Our theoretical results address the optimized ridgelet-based hidden-node sampling task for discretized single-hidden-layer neural networks. They do not directly imply an efficient lottery-ticket discovery method for general deep neural networks. Our theoretical guarantees apply to the finite-field setting and do not establish approximation guarantees for regression on continuous domains.

The polynomial-time guarantees hold in the regime where the relevant parameters are polynomially bounded. In particular, the runtime depends on the number of training samples M, the discretization parameter P through polylogarithmic factors, the inverse smoothing parameter Δ^{-1}, the weight-scale parameter γ, and the sampling accuracy through $\log(1/\delta)$. The runtime bound of our classical sampler has a worst-case quadratic dependence on M, up to logarithmic factors.

The numerical experiments are intended as controlled demonstrations of the sampling procedure. They use synthetic data for regression tasks. Therefore, the experiments do not establish performance on real-world datasets or deep neural-network architectures. We do not provide a general approximation guarantee for discretizing continuous regression problems into the finite-field setting. Extending the method beyond the discretized ridgelet setting, in particular to deep neural-network architectures, remains an important direction for future work.

F.2 Broader Impact

This work is a foundational study of a sampling procedure for discretized single-hidden-layer neural-network models. It does not introduce a deployed system, a pretrained model, a generative model, or a dataset involving human subjects.

We do not identify any direct societal impacts, positive or negative, from this work as presented. Any societal implications would likely arise only from application-specific adaptations of the method, which should be evaluated in the context of those downstream applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction identify the contribution as a quantum-inspired classical algorithm for the optimized hidden-node sampling task used in ridgelet-based sparse-subnetwork construction. The scope is stated as discretized single-hidden-layer neural networks, and Appendix F further clarifies the main limitations and scope of the claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix F explicitly discusses the main limitations, including the restriction to discretized single-hidden-layer neural networks, the dependence on the problem parameters M, P, Δ, γ, and δ, and the use of synthetic regression data rather than real-world datasets or deep neural-network architectures.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions of the main classical and improved quantum sampling results are stated in Section 2 and Theorems 2 and 3. The key inverse-decomposition lemma and complete proofs are provided in Appendices B, C, and D.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 and Appendix E describe the finite-domain data generation, target function, activation and ridgelet functions, regularization convention, sampling procedures, node deduplication, and ridge-regression fitting. The experiments use synthetic data generated from the stated probability distributions, so no external dataset is required for reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The experiments use only synthetic data. The data-generating distribution, target function, and experimental settings are fully described in Appendix E. The same procedures are implemented in the submitted supplementary code. We provide a requirements file and a README with the commands needed to reproduce the figures in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix E provide the data-generation distribution, target function, activation and ridgelet functions, regularization parameters, smoothing parameter, sampling procedures, and ridge-regression fitting step.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report the number of repetitions used in our experiments. Figure 1 shows the mean over 50 executions with error bars representing the standard error of the mean, and Figure 2 shows the mean over 20 executions with 95% confidence intervals.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E reports the computational environment used for the numerical experiments: a MacBook Pro with an Apple M4 Pro CPU and 48 GB unified memory, using CPU execution with Python and NumPy. The runtime experiments report wall-clock times measured by `time.perf_counter()`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work conforms to the NeurIPS Code of Ethics since the work uses only synthetic finite-domain experiments, and it does not involve human subjects, personal data, private data, deployed systems, or safety-critical applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Appendix F discusses broader-impact considerations. The work is a foundational study of a sampling procedure for discretized single-hidden-layer neural-network models; it does not introduce a pretrained model, a generative model, a dataset involving human subjects, or a deployed system. We do not identify direct societal impacts, positive or negative, from the work as presented.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a pretrained model, generative model, scraped dataset, human-subject dataset, or other artifact with a high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The experiments do not use external datasets, pretrained models, or third-party research assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the supplementary code files. The algorithms are explained in the paper, and the supplementary code provides implementations together with a README and requirements file.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not include crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM assistance was not used as an important, original, or non-standard component of the core method, theory, or experimental protocol. It was used only as auxiliary assistance for writing, editing, formatting, and implementation support for experiment scripts. All algorithms, proofs, code, and experimental settings were specified, checked, and validated by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.