Quantum Ridgelet Transform: Winning Lottery Ticket of Neural Networks with Quantum Computation

Hayata Yamasaki [1] **Sathyawageeswar Subramanian** [2] **Satoshi Hayakawa** [3] **Sho Sonoda** [4]

Abstract

A significant challenge in the field of quantum machine learning (QML) is to establish applications of quantum computation to accelerate common tasks in machine learning such as those for neural networks. Ridgelet transform has been a fundamental mathematical tool in the theoretical studies of neural networks, but the practical applicability of ridgelet transform to conducting learning tasks was limited since its numerical implementation by conventional classical computation requires an exponential runtime $\exp(O(D))$ as data dimension D increases. To address this problem, we develop a quantum ridgelet transform (QRT), which implements the ridgelet transform of a quantum state within a linear runtime $O(D)$ of quantum computation. As an application, we also show that one can use QRT as a fundamental subroutine for QML to efficiently find a sparse trainable subnetwork of large shallow wide neural networks without conducting large-scale optimization of the original network. This application discovers an efficient way in this regime to demonstrate the lottery ticket hypothesis on finding such a sparse trainable neural network. These results open an avenue of QML for accelerating learning tasks with commonly used classical neural networks.

1. Introduction

Quantum machine learning (QML) is an emerging field of research to take advantage of quantum computation for accelerating machine-learning tasks (Biamonte et al., 2017; Ciliberto et al., 2018; Schuld & Petruccione, 2021). Quantum computation can achieve significant speedups compared to the best existing algorithms with conventional classical computation in solving various computational tasks (Nielsen & Chuang, 2011; de Wolf, 2019), such as Shor's algorithm for integer factorization (Shor, 1997). QML indeed has advantages in learning data obtained from quantum states (Sweke et al., 2021; Huang et al., 2021; 2022; Chen et al., 2022a), yet machine learning commonly deals with classical data rather than quantum states. For a classical dataset constructed carefully so that its classification reduces to a variant of Shor's algorithm, QML achieves the classification superpolynomially faster than classical algorithms (Liu et al., 2021); however, the applicability of such QML to practical datasets has been unknown. Meanwhile, motivated by the success of neural networks (Goodfellow et al., 2016), various attempts have been made to apply quantum computation to more practical tasks for neural networks. For example, one widely studied approach in QML is to use parameterized quantum circuits, often called "quantum neural networks", as a potential substitute for conventional classical neural networks; however, problematically, the parameterized quantum circuits do not successfully emulate essential components of the neural networks, e.g., perceptrons and nonlinear activation functions, due to linearity of the transformation implemented by the quantum circuits (Schuld & Petruccione, 2021). Thus, a significant challenge in QML has been to develop a novel technique to bridge the gap between quantum computation and classical neural networks, so as to clarify what advantage QML could offer on top of the empirically proven merit of the classical neural networks.

To address this challenge, we here develop a fundamental quantum algorithm for making the tasks for classical neural networks more efficient, based on ridgelet transform. Ridgelet transform, one of the well-studied integral transforms in signal processing, is a fundamental mathematical tool for studying neural networks in the over-parameterized regime (Murata, 1996; Candes, 1998; Rubin, 1998; Starck et al., 2010; Sonoda et al., 2021; 2022a;b). Let $f : \mathbb{R}^D \to \mathbb{R}$ denote a function with D-dimensional input, to be learned with a neural network. For an activation function $g : \mathbb{R} \to \mathbb{R}$ such as the rectified linear unit (ReLU), a shallow feedforward neural network with a single hidden layer is represented by $f(\boldsymbol{x}) \approx \sum_{n=1}^{N} w_n g(\boldsymbol{a}_n^\top \boldsymbol{x} - b_n)$, where N is the number of nodes in the hidden layer, and w_n is the

[1]The University of Tokyo [2]University of Warwick [3]University of Oxford [4]RIKEN AIP. Correspondence to: Hayata Yamasaki <hayata.yamasaki@phys.s.u-tokyo.ac.jp>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

weight of the map $g(\boldsymbol{a}_n^\top \boldsymbol{x} - b_n)$ parameterized by $(\boldsymbol{a}_n, b_n)$ at node $n \in \{1, \ldots, N\}$ (Goodfellow et al., 2016). In the over-parameterized (continuous) limit $N \to \infty$, the representation simplifies into an **integral representation** of the neural network (Barron, 1993; Murata, 1996; Candes, 1998; Sonoda & Murata, 2017), i.e.,

$$f(\boldsymbol{x}) = S[w](\boldsymbol{x}) := \int_{\mathbb{R}^D \times \mathbb{R}} d\boldsymbol{a}\, db\, w(\boldsymbol{a}, b) g(\boldsymbol{a}^\top \boldsymbol{x} - b), \quad (1)$$

where $(\boldsymbol{a}, b)$ runs over all possible parameters in the continuous space, and $w : \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}$ at each $(\boldsymbol{a}, b)$ corresponds to the weight w_n at the node n with parameter $(\boldsymbol{a}_n, b_n) = (\boldsymbol{a}, b)$. With a ridgelet function $r : \mathbb{R}^D \to \mathbb{R}$ that we appropriately choose corresponding to g, the D-dimensional **ridgelet transform** $R[f]$ is defined as an inverse transform of $S[w]$ in (1), characterizing a weight w to represent f, given by

$$w(\boldsymbol{a}, b) = R[f](\boldsymbol{a}, b) := \int_{\mathbb{R}^D} d\boldsymbol{x}\, f(\boldsymbol{x}) r(\boldsymbol{a}^\top \boldsymbol{x} - b). \quad (2)$$

A wide class of function f is known to be representable as (1); moreover, if g and r satisfy a certain admissibility condition, we can reconstruct f from the ridgelet transform of f, i.e., $f \propto S[R[f]]$, up to a normalization factor (Sonoda & Murata, 2017). For theoretical analysis, an essential benefit of the integral representation is to simplify the analysis by the linearity; that is, we can regard (1) as the linear combination of an non-orthogonal over-complete basis of functions, i.e., $\{g(\boldsymbol{a}^\top \boldsymbol{x} - b) : (\boldsymbol{a}, b) \in \mathbb{R}^D \times \mathbb{R}\}$, with weight $w(\boldsymbol{a}, b)$ given by the ridgelet transform of f.

Progressing beyond using the ridgelet transform for theoretical analysis, our key idea is to study its use for conducting tasks for neural networks. However, D-dimensional ridgelet transform has been computationally hard to use in practice since the existing algorithms for ridgelet transform with conventional classical computation require $\exp(O(D))$ runtime as D increases (Do & Vetterli, 2003; Carre & Andres, 2004; Helbert et al., 2006; Sonoda & Murata, 2014). After all, the D-dimensional ridgelet transform is a transform of D-dimensional functions in an $\exp(O(D))$-size space (see Sec. 2 for detail), and classical algorithms for such transforms conventionally need $\exp(O(D))$ runtime; e.g., fast Fourier transform may be a more established transform algorithm but still needs $O(n \log(n)) = \exp(O(D))$ runtime for the space of size $n = \exp(O(D))$. To solve these problems, we discover that we can employ quantum computation. Our results are as follows.

1. (Sec. 2) To make exact implementation of ridgelet transform possible for computer with a finite number of bits and quantum bits (qubits), we formulate a new discretized version of ridgelet transform, which we call **discrete ridgelet transform**. We prove that our discretized ridgelet transform can be used for exactly representing any function on the discretized domain.

2. (Sec. 3) We develop a quantum algorithm to apply the D-dimensional discrete ridgelet transform to a quantum state of $O(D)$ qubits, i.e., a state in an $\exp(O(D))$-dimensional space, only within linear runtime $O(D)$. We call this quantum algorithm **quantum ridgelet transform (QRT)**. QRT is exponentially faster in D than the $\exp(O(D))$ runtime of the best existing classical algorithm for ridgelet transform in the $\exp(O(D))$-size space, in the same spirit as **quantum Fourier transform (QFT)** (Coppersmith, 1994) being exponentially faster than the corresponding classical algorithm of fast Fourier transform.

3. (Sec. 4) As an application, we demonstrate that we can use QRT to learn a sparse representation of an unknown function f by sampling a subnetwork of a shallow wide neural network to approximate f well. We analytically show the advantageous cases of our algorithm and also conduct a numerical simulation to support the advantage. This application is important as a demonstration of the **lottery ticket hypothesis** (Frankle & Carbin, 2019), as explained in the following.

Contribution to QML with neural networks State-of-the-art neural networks have billions of parameters to attain high learning accuracy, but such large-scale networks may be problematic for practical use, e.g., with mobile devices and embedded systems. Pruning techniques for neural networks gain growing importance in learning with neural networks. The lottery ticket hypothesis by Frankle & Carbin (2019) claims that, in such a large-scale neural network, one can find a sparse trainable subnetwork. However, it is computationally demanding to search for the appropriate subnetwork in the large-scale neural network.

To apply QML to this pruning problem, our idea is to use QRT for preparing a quantum state so that, by measuring the state, we can sample the parameters of the important nodes for the subnetwork with high probability. To make this algorithm efficient, we never store all parameters of the large original neural network in classical memory but represent them by the amplitude of the quantum state prepared directly from given data. Conventionally, quantum computation can use QFT to achieve superpolynomial speedups over classical computation for various search problems (Simon, 1994; Shor, 1997; Bernstein & Vazirani, 1997; Yamakawa & Zhandry, 2022). By contrast, we make quantum computation applicable to searching in the parameter space of neural networks, by developing QRT to be used in place of QFT.

Consequently, our results show that QRT can be used as a fundamental subroutine for QML to accelerate the tasks for the classical neural networks. A potential drawback may be that our current technique is designed simply for the shallow neural networks with a single hidden layer; however, studies of shallow neural networks capture various essential features of neural networks. We leave the generalization to deep neural networks for future research, but our development opens a promising route in this direction. More specifically, if one develops, e.g., a theory to generalize ridgelet transform to the analysis of deep neural networks, our results are expected to offer fundamental techniques for obtaining similar applications to deep neural networks. The significance of our results is to bridge the gap between a theoretical development in analyzing neural networks based on ridgelet transform and the algorithmic techniques in quantum computation for accelerating the machine learning tasks on top of the theoretical development. The essential benefit of ridgelet transform is to provide a closed formula for the weights of the nodes in the hidden layer of a shallow neural network to represent the function to be learned in the over-parameterized (continuous) limit. For deep neural networks, such a closed formula may still be unknown at the moment, but in recent years, significant theoretical progress has been made in the analysis of deep neural networks in the over-parameterized regime. Given such progress, our contributions on the quantum side, if combined with further development in the theoretical analysis of deep neural networks, open a fundamental way to explore the applications in this direction.

2. Discrete Ridgelet Transform

2.1. Formulation of Discrete Ridgelet Transform

In this section, we formulate the **discrete ridgelet transform**. Then, we also derive a **Fourier slice theorem** that characterizes our discrete ridgelet transform using Fourier transform. Although multiple definitions of discrete versions of ridgelet transform have been proposed, none of them has such Fourier expression (Do & Vetterli, 2003; Carre & Andres, 2004; Helbert et al., 2006). By contrast, the significance of the Fourier slice theorem is that it makes the ridgelet analysis tractable with the well-established techniques for the Fourier transform, which we will use in Sec. 3 for constructing the quantum algorithm as well.

Our formulation assumes the following.

- Since computers using a finite number of bits and qubits cannot exactly deal with real number, we use a finite set $\mathbb{Z}_P := \{0, 1, \ldots, P-1\}$ in place of the set of real number $\mathbb{R}$, where P is a precision parameter representing the cardinality of $\mathbb{Z}_P$. This is conventional in data representation; e.g., for a gray scale image, we may use 8 bits $\{0, 1, \ldots, 2^8-1\}$ to represent the intensity of each pixel. In our setting, we can make P larger to achieve better precision in the data representation; e.g., for a more precise representation of the gray scale image, we may use 16 bits $\{0, 1, \ldots, 2^{16}-1\}$ in place of the 8 bits for each pixel. For this improvement of the precision, we may normalize the data by rescaling while the intervals in the discretization are fixed to 1 for simplicity of presentation. When we write a sum, $\boldsymbol{x}$, $\boldsymbol{a}$, and $\boldsymbol{u}$ run over $\mathbb{Z}_P^D$, and y, b, and v over $\mathbb{Z}_P$ unless specified otherwise. Let $\mathcal{F}_D$ and $\mathcal{F}_1$ denote the D-dimensional and 1-dimensional discrete Fourier transforms on $\mathbb{Z}_P^D$ and $\mathbb{Z}_P$, respectively, i.e.,

$$\mathcal{F}_D[f](\boldsymbol{u}) := P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) \mathrm{e}^{\frac{-2\pi i \boldsymbol{u}^\top \boldsymbol{x}}{P}}, \quad (3)$$

$$\mathcal{F}_1[g](v) := P^{-\frac{1}{2}} \sum_{b} r(b) \mathrm{e}^{\frac{-2\pi i v b}{P}}. \quad (4)$$

- We assume in the following that P is taken as a prime number, considering $\mathbb{Z}_P$ as a finite field. This is not a restrictive assumption in achieving the better precision since we can take an arbitrarily large prime number as P. For example, the maximum of 32-bit signed integers $P = 2^{31}-1$ can be used.

- An activation function $g : \mathbb{R} \to \mathbb{R}$ is assumed to be normalized as

$$\sum_b g(b) = 0, \quad \|g\|_2^2 := \sum_b |g(b)|^2 = 1. \quad (5)$$

Indeed, any non-constant function, such as ReLU, can be used as g with normalization by adding and multiplying appropriate constants.

- Corresponding to g, we choose a ridgelet function $r : \mathbb{R} \to \mathbb{R}$ in such a way that g and r satisfy an admissibility condition

$$C_{g,r} := \sum_v \mathcal{F}_1[g](v) \overline{\mathcal{F}_1[r](v)} \neq 0, \quad (6)$$

where $\overline{\cdots}$ denotes the complex conjugate. We also normalize r as $\|r\|_2^2 = 1$. Note that it is conventional to choose $r = g$, leading to $C_{g,g} = \|\mathcal{F}_1[g]\|_2^2 = \|g\|_2^2 = 1$, while our setting allows any non-unique choice of r satisfying (6) in general.

Then, by replacing the integral over $\mathbb{R}$ in S of (1) and R of (2) with the finite sum over $\mathbb{Z}_P$, we correspondingly define the **discretized neural network** and the **discrete ridgelet transform** of $f : \mathbb{R}^D \to \mathbb{R}$ as, respectively,

$$\mathcal{S}[w](\boldsymbol{x}) := P^{-\frac{D}{2}} \sum_{\boldsymbol{a},b} w(\boldsymbol{a}, b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P), \quad (7)$$

$$\mathcal{R}[f](\boldsymbol{a}, b) := P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) r((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P), \quad (8)$$

where $P^{-\frac{D}{2}}$ represents a normalization constant. Note that the discretized neural network $\mathcal{S}[w](\boldsymbol{x})$ has discrete parameters $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ for nodes in the hidden layer, but the weights $w(\boldsymbol{a}, b) \in \mathbb{R}$ of the nodes are real numbers.

The following theorem, **Fourier slice theorem**, characterizes the discrete ridgelet transform $R[f]$ in terms of Fourier transforms of f and h. In the case of continuous ridgelet transform, ridgelet analysis can be seen as a form of wavelet analysis in the Radon domain, which combines wavelet and Radon transforms (Fadili & Starck, 2012). The Fourier slice theorem for the continuous ridgelet transform follows from that for continuous Radon transform. However, problematically, discrete versions of the Radon transforms are nonunique and usually more involved, not necessarily having exact expression in terms of discrete Fourier transform (Beylkin, 1987; Kelley & Madisetti, 1993; Götz & Druckmüller, 1996; Brady, 1998; Boag et al., 2000; Brandt et al., 2000; Press, 2006). As a result, the existing definitions of discrete versions of ridgelet transform have no such Fourier expression either (Do & Vetterli, 2003; Carre & Andres, 2004; Helbert et al., 2006). By contrast, we here show that our formulation of the discrete ridgelet transform $\mathcal{R}[f]$ has the following exact characterization in the Fourier transform. See Appendix A for proof.

Theorem 2.1 (Fourier slice theorem for discrete ridgelet transform)**.** *For any function $f : \mathbb{R}^D \to \mathbb{R}$ and any point $\boldsymbol{a} \in \mathbb{Z}_P^D, v \in \mathbb{Z}_P$ in the discretized space, it holds that*

$$\mathcal{F}_1[\mathcal{R}[f](\boldsymbol{a}, \cdot)](v) = \mathcal{F}_D[f](v\boldsymbol{a} \bmod P)\, \overline{\mathcal{F}_1[r](v)}. \quad (9)$$

2.2. Exact Representation of Functions as Neural Networks

Using the discrete ridgelet transform in Sec. 2.1, we here show that any function f on the discretized domain has an **exact representation** in terms of a shallow neural network with a finite number of parameters in the discretized space. In the continuous case, any square-integrable function f is represented as the continuous limit of the shallow neural networks, i.e., $f = S[w]$ in (1), with the weight given by the ridgelet transform $w \propto R[f]$ (Sonoda & Murata, 2017). With discretization, it is nontrivial to show such an exact representation due to finite precision in discretizing the real number. Nevertheless, we here show that any function $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{Z}_P^D$ can be exactly represented as $f(\boldsymbol{x}) = \mathcal{S}[w](\boldsymbol{x})$ as well, with the weight given by our formulation of the discrete ridgelet transform $w \propto \mathcal{R}[f]$, which we may call exact representation as a discretized neural network.

In particular, the following theorem shows that any D-dimensional real function f on the discrete domain can be exactly represented as a linear combination of non-orthogonal basis functions $\{g((\boldsymbol{a}^\top\boldsymbol{x}-b) \bmod P) : (\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P\}$ with coefficients given by $\mathcal{R}[f]$. Due to the non-orthogonality, the coefficients may not be unique, and different choices of the ridgelet function r in (8) lead to different $\mathcal{R}[f]$ while any of the choices can exactly reconstruct f. The proof is based on the Fourier slice theorem in Theorem 2.1, crucially using the existence of an inverse element for each element of a finite field $\mathbb{Z}_P$; thus, it is essential to assume that P is a prime number. See Appendix B for proof.

Theorem 2.2 (Exact representation of function as discretized neural network)**.** *For any function $f : \mathbb{R}^D \to \mathbb{R}$ and any point $\boldsymbol{x} \in \mathbb{Z}_P^D$ in the discretized domain, we have*

$$f(\boldsymbol{x}) = C_{g,r}^{-1}\, \mathcal{S}[\mathcal{R}[f]](\boldsymbol{x}), \quad (10)$$

where $C_{g,r}$ is a constant defined in (6)*.*

3. Quantum Ridgelet Transform

In this section, we introduce **quantum ridgelet transform (QRT)**, an efficient quantum algorithm for implementing the discrete ridgelet transform formulated in Sec. 2. In various quantum algorithms, we may use quantum Fourier transform (QFT) as a fundamental subroutine. In addition to QFT, various discrete transforms can be efficiently implemented with quantum computation, such as wavelet transform (Hoyer, 1997; Fijany & Williams, 1998; Labunets et al., 2001a; Argüello, 2009; Taha, 2016; Li et al., 2019; 2022), Radon transform (Ma et al., 2022), fractional Walsh transform (Labunets et al., 2001b), Hartley transform (Tseng & Hwang, 2004), and curvelet transform (Liu, 2009). However, the existing discrete versions of ridgelet transform (Do & Vetterli, 2003; Carre & Andres, 2004; Helbert et al., 2006) were lacking implementation by quantum computation. In contrast, our QRT opens a way to use the discrete ridgelet transform as a fundamental subroutine for QML to deal with tasks for classical neural networks.

Basic notions and notations of quantum computation to describe our quantum algorithms are summarized in Appendix C. In classical computation, we may use $\lceil \log_2(P) \rceil$ bits for representing $\mathbb{Z}_P$, where $\lceil x \rceil$ denotes the ceiling function, i.e., the smallest integer that is not smaller than x. The quantum algorithm uses a $\lceil \log_2(P) \rceil$-qubit quantum register for $\mathbb{Z}_P$. This quantum register for $\mathbb{Z}_P$ is represented as a $2^{\lceil \log_2(P) \rceil}$-dimensional complex vector space $\mathcal{H}_P \coloneqq (\mathbb{C}^2)^{\otimes \lceil \log_2(P) \rceil}$. Using the conventional bra-ket notation, each state in the standard orthonormal basis of the registers is written as a ket (i.e., a vector) $|\boldsymbol{x}\rangle \in \mathcal{H}_P^{\otimes D}$ for representing $\boldsymbol{x} \in \mathbb{Z}_P^D$ and $|\boldsymbol{a}, b\rangle \coloneqq |\boldsymbol{a}\rangle \otimes |b\rangle \in \mathcal{H}_P^{\otimes D} \otimes \mathcal{H}_P$ for $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$, respectively.

The task of QRT is to transform a given unknown quantum state $|\psi\rangle = \sum_{\boldsymbol{x}} \psi(\boldsymbol{x}) |\boldsymbol{x}\rangle$ into $\boldsymbol{R}|\psi\rangle =$

$\sum_{\boldsymbol{a},b} \mathcal{R}[\psi](\boldsymbol{a},b) \ket{\boldsymbol{a},b}$, where $\boldsymbol{R}$ is a matrix given by

$$\boldsymbol{R} \coloneqq P^{-\frac{D}{2}} \sum_{\boldsymbol{x},\boldsymbol{a},b} r((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) \ket{\boldsymbol{a},b}\bra{\boldsymbol{x}}. \quad (11)$$

Under the assumptions in Sec. 2.1, $\boldsymbol{R}$ becomes an isometry matrix and can be implemented by a quantum circuit for our quantum algorithm, as shown below. Along with the assumptions in Sec. 2.1, we use the following assumption to bound the runtime of our algorithm.

- We choose the ridgelet function $r : \mathbb{R} \to \mathbb{R}$ in such a way that a quantum state representing r by its amplitude $\ket{r} \coloneqq \sum_y r(y)\ket{y}$ can be prepared efficiently in runtime $O(\mathrm{polylog}(P))$. This assumption is not restrictive since we can use the quantum algorithm by Grover & Rudolph (2002) to meet the assumption for representative choices of r that are integrable efficiently, such as ReLU and tanh.

Algorithm 1 shows our quantum algorithm for QRT. We construct the algorithm by implementing the discrete Fourier transform in the Fourier slice theorem (Theorem 2.2) by QFT. QFT applies a unitary matrix representing discrete Fourier transform

$$\boldsymbol{F}_P \coloneqq \sum_{v,b} P^{-\frac{1}{2}} \mathrm{e}^{\frac{-2\pi \mathrm{i} vb}{P}} \ket{v}\bra{b} \quad (12)$$

to a given quantum state of $\mathcal{H}_P$ within runtime $O(\mathrm{polylog}(P))$ (Mosca & Zalka, 2004). The following theorem shows that this speedup is also the case in QRT compared to classical algorithms for computing ridgelet transform. See Appendix C for proof.

Theorem 3.1 (Runtime of quantum ridgelet transform)**.** *The runtime of QRT in Algorithm 1 is*

$$O(D \times \mathrm{polylog}(P)). \quad (13)$$

The advantage of QRT is its linear runtime $O(D)$ in the data dimension D, which is exponentially faster than the best existing classical algorithm for ridgelet transform in the $\exp(O(D))$-size space requiring $\exp(O(D))$ runtime. This advantage is in the same spirit as the QFT being exponentially faster than the corresponding classical algorithm for fast Fourier transform in the spaces of the same size. In Sec. 4, we will further clarify that QRT has an application to accelerate the task of finding the winning ticket of neural networks.

Algorithm 1 Quantum ridgelet transform (QRT).

```
Require: A given input state $\ket{\psi} = \sum_{\boldsymbol{x}} \psi(\boldsymbol{x})\ket{\boldsymbol{x}}$, the
    ridgelet function $r$ satisfying the assumptions in Sec. 3.
Ensure: Output $\boldsymbol{R}\ket{\psi} = \sum_{\boldsymbol{a},b} \mathcal{R}[\psi](\boldsymbol{a},b)\ket{\boldsymbol{a},b}$ in (11)
    within runtime $O(D \times \mathrm{polylog}(P))$ as in Theorem 3.1.
 1: Given $\ket{\psi}$, add an auxiliary register $\mathcal{H}_P$ prepared in
    $\sum_b r(b)\ket{b}$, to obtain $\sum_{\boldsymbol{x},b} \psi(\boldsymbol{x})\ket{\boldsymbol{x}} \otimes r(b)\ket{b}$.
 2: Apply $D$-dimentional QFT $\boldsymbol{F}_P^{\otimes D}$ and 1-dimensional in-
    verse QFT $\boldsymbol{F}_P^\dagger$ to the first and second quantum registers,
    respectively, to transform $\sum_{\boldsymbol{x},b} \psi(\boldsymbol{x})\ket{\boldsymbol{x}} \otimes r(b)\ket{b}$ into
        $$\sum_{\boldsymbol{a}' \in \mathbb{Z}_P^D} \sum_{v \in \mathbb{Z}_P} \mathcal{F}_D[\psi](\boldsymbol{a}')\ket{\boldsymbol{a}'} \otimes \overline{\mathcal{F}_1[r](v)}\ket{v}. \quad (14)$$
 3: Perform arithmetics $\ket{\boldsymbol{a}'} \mapsto \ket{v^{-1}\boldsymbol{a}' \bmod P}$ on the
    first register by controlled gates that are controlled by
    the state $\ket{v}$ of the second, to obtain
        $$\sum_{v \in \mathbb{Z}_P \setminus \{0\}} \sum_{\boldsymbol{a}' \in \mathbb{Z}_P^D} \mathcal{F}_D[\psi](\boldsymbol{a}')\ket{v^{-1}\boldsymbol{a}' \bmod P} \otimes \overline{\mathcal{F}_1[r](v)}\ket{v}$$
        $$= \sum_{\boldsymbol{a},v} \mathcal{F}_D[\psi](v\boldsymbol{a} \bmod P)\overline{\mathcal{F}_1[r](v)}\ket{\boldsymbol{a}} \otimes \ket{v}, \quad (15)$$
    where $v^{-1} \in \mathbb{Z}_P$ is the inverse of $v$ in the finite field
    $\mathbb{Z}_P$, $\boldsymbol{a} \coloneqq v^{-1}\boldsymbol{a}'$, and $\bmod P$ for $\mathbb{Z}_P^D$ is taken element-
    wise.
 4: Apply inverse QFT $\boldsymbol{F}_P^\dagger$ to the second quantum register,
    which yields $\boldsymbol{R}\ket{\psi}$ due to Theorem 2.1.
 5: Return $\boldsymbol{R}\ket{\psi}$.
```

4. Application of Quantum Ridgelet Transform to Lottery Ticket Hypothesis

4.1. Setting for Winning Ticket of Neural Networks

In this section, as an application of quantum ridgelet transform (QRT) in Sec. 3, we propose an algorithm for finding a sparse subnetwork approximating a large neural network efficiently by quantum computation, based on the **lottery ticket hypothesis** on neural networks. The lottery ticket hypothesis by Frankle & Carbin (2019) claims that a randomly-initialized fully-connected neural network contains a subnetwork that is initialized in such a way that, when trained in isolation, it can match the accuracy of the original network after training for at most the same number of iterations. This hypothesis has been confirmed numerically in various settings. The theoretical analysis of deep neural networks is inevitably hard in general, and studies of shallow neural networks are also important for capturing essences of neural networks, which we here consider in a setting of regression from given data as described in the following.

For $D \in \{1,2,\ldots\}$ with a fixed prime number P, we consider a family of problems to approximate an unknown function $f^{(D)} : \mathbb{R}^D \to \mathbb{R}$ by a shallow neural network, i.e.,

$$\hat{f}^{(D)}(\boldsymbol{x}) \coloneqq \sum_{n=1}^{N} w_n g((\boldsymbol{a}_n^\top \boldsymbol{x} - b_n) \bmod P). \quad (16)$$

Let $p_{\mathrm{data}}^{(D)}$ be a probability mass function for the input data, which is assumed to be supported on $\mathbb{Z}_P^D$. Suppose that we are given M input-output pairs of examples $(\boldsymbol{x}_1, y_1 = f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_M, y_M = f(\boldsymbol{x}_M)) \in \mathbb{Z}_P^D \times \mathbb{R}$. Let $\hat{p}_{\mathrm{data}}^{(D)}$ denote the empirical distribution of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$. Given $\epsilon > 0$, we will analyze empirical risk minimization (Bach, 2021), i.e., minimization of the empirical risk $\sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}^{(D)}(\boldsymbol{x})|f^{(D)}(\boldsymbol{x}) - \hat{f}^{(D)}(\boldsymbol{x})|^2$ to $O(\epsilon)$. If obvious, we may omit D in superscripts; e.g., we may write $f^{(D)}$ as f.

The setting of our analysis, along with the assumptions in Sec. 3, is as follows. Based on the exact representation of $f(\boldsymbol{x})$ in terms of the neural network $\mathcal{S}[w](\boldsymbol{x})$ in Theorem 2.2, we can approximate f by a neural network

$$f(\boldsymbol{x}) \approx \mathcal{S}[w_\lambda^*](\boldsymbol{x}) = \sum_{\boldsymbol{a},b} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a}, b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P), \tag{17}$$

where w_λ^* is the optimal solution of the ridge regression with the empirical distribution, i.e.,

$$w_\lambda^*(\boldsymbol{a}, b) := \arg\min_w \{\tilde{J}(w)\}, \tag{18}$$

$\tilde{J}(w) := J(w) + \lambda\Omega(w)$, $J(w) := \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})|f(\boldsymbol{x}) - \mathcal{S}[w](\boldsymbol{x})|^2$, $\Omega(w) := \|P^{-\frac{D}{2}} w\|_2^2 = \sum_{\boldsymbol{a},b} |P^{-\frac{D}{2}} w(\boldsymbol{a}, b)|^2$, and $\lambda > 0$ is a hyperparameter for regularization. Learning a general class of function $f^{(D)}$ on $\mathbb{Z}_P^D$ would be inevitably demanding as its representation would require $\exp(O(D))$ parameters to specify the values $f^{(D)}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{Z}_P^D$ in the worst case. We have shown such a general representation in Theorem 2.2. By contrast, our goal here is to achieve the approximation feasibly with much fewer parameters, using a subnetwork of the large original network $\mathcal{S}[w_\lambda^*](\boldsymbol{x})$.

To this goal, recall that it is conventional in statistical learning theory to consider a reasonably restricted class of functions, e.g., those with bounded norms (Bach, 2021); correspondingly, we work on a setting where the norm $\|P^{-\frac{D}{2}} w_\lambda^*\|_1 := \sum_{\boldsymbol{a},b} |P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a}, b)|$ of the weights for representing $f^{(D)}$ should be bounded even on the large scales $D \to \infty$. In particular, let $((\boldsymbol{a}_j, b_j) \in \mathbb{Z}_P^D \times \mathbb{Z}_P : j \in \{1, \ldots, P^{D+1}\})$ denote a sequence of parameters of all nodes in the hidden layer of $\mathcal{S}[w_\lambda^*](\boldsymbol{x})$ aligned in the descending order of w_λ^*, i.e., $|w_\lambda^*(\boldsymbol{a}_1, b_1)| \geqq |w_\lambda^*(\boldsymbol{a}_2, b_2)| \geqq \cdots$. These nodes of $\mathcal{S}[w_\lambda^*]$ are ordered in the same way; i.e., the weight of the jth node is $w_\lambda^*(\boldsymbol{a}_j, b_j)$. Then, we assume the following.

- Following the convention of assumptions in the previous works by, e.g., Donoho (1993); Hayakawa & Suzuki (2020), we assume that there exist constants $\alpha, \beta > 0$ such that it holds uniformly for any D and j that
$$|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a}_j, b_j)| \leqq \alpha j^{-(1+\beta)}, \tag{19}$$
which specifies the decay rate for large j, leading to $\|P^{-\frac{D}{2}} w_\lambda^*\|_1 \leqq \sum_{j=1}^{\infty} \alpha j^{-(1+\beta)} \leqq \alpha + \int_1^\infty \frac{\alpha}{x^{1+\beta}} dx = \alpha + \frac{\alpha}{\beta} < \infty$. We also write the L^2 norm as
$$\gamma := \|P^{-\frac{D}{2}} w_\lambda^*\|_2^2, \tag{20}$$
which is upper bounded by $\gamma \leqq \sum_{j=1}^{\infty} \alpha^2 j^{-2(1+\beta)} \leqq \alpha^2 + \int_1^\infty \frac{\alpha^2}{x^{2(1+\beta)}} dx = \alpha^2 + \frac{\alpha^2}{1+2\beta}$. Functions $(f^{(D)} : D = 1, 2, \ldots)$ satisfying (19) are called (α, β)-class functions, which are to be learned in our setting.

For given $\epsilon > 0$, our analysis focuses on the task of finding a sparse representation $\hat{f}$ to approximate $\mathcal{S}[w_\lambda^*]$ up to ϵ, i.e.,

$$\sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})|\mathcal{S}[w_\lambda^*](\boldsymbol{x}) - \hat{f}(\boldsymbol{x})|^2 = O(\epsilon), \tag{21}$$

with keeping the number of nodes N in the hidden layer of $\hat{f}$ in (16) as small as possible. Following the conventional prescription in the statistical learning theory (Bach, 2021), we assume that we appropriately choose

$$\lambda \approx \mathrm{poly}(\epsilon), \tag{22}$$

so that (21) leads to $\sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x})|f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})|^2 = O(\epsilon)$, achieving the empirical risk minimization with $\hat{f}$. Note that if we fix D, then for any f, we may be able to find sufficiently large α and small β to meet (19), but our analysis will show that assuming smaller α and larger β guarantees smaller N to achieve (21) for the (α, β)-class functions for arbitrary D.

4.2. Quantum Algorithm for Sampling from Optimized Probability Distribution

We here construct a quantum algorithm for sampling from an **optimized probability distribution** (defined later as $p_{\lambda,\Delta}^*(\boldsymbol{a}, b)$ in (23)) of parameters of nodes in the hidden layer of the large original network $\mathcal{S}[w_\lambda^*]$ in (17), which we can use for efficiently finding a sparse subnetwork of $\mathcal{S}[w_\lambda^*]$ to approximate f well. The original network $\mathcal{S}[w_\lambda^*]$ has $\exp(O(D))$ nodes to represent any function f, as with Theorem 2.2. By contrast, studies on the lottery ticket hypothesis provide numerical evidences that f in practice can usually be approximated by a sparse subnetwork with much fewer parameters. One existing way to find such a subnetwork is to train the overall large network and then perform masking to eliminate the low-weight nodes while keeping those with higher weights (Frankle & Carbin, 2019). However, this approach is inefficient since one needs large-scale optimization to train the large original network before the pruning. Then, more recent studies by Lee et al. (2019); Zhou et al. (2019); Ramanujan et al. (2020); Malach et al. (2020); Orseau et al. (2020); Pensia et al. (2020); Tanaka et al. (2020); Wortsman et al. (2020); Wang et al. (2020a;b); Frankle et al. (2021); Chen et al. (2022b) have suggested that one should be able to find the subnetwork only by pruning the initial network directly, even without the optimization for training. Still, to perform this pruning appropriately, one needs to perform a large-scale search for the subnetwork within the parameter space of the large original neural network. As D increases, it would become infeasible to deal with the large original network for training or searching as long as we use the existing methods based on classical computation.

To address this problem, our key idea is to represent the weights of the $\exp(O(D))$ nodes in the hidden layer of the neural network $\mathcal{S}[w_\lambda^*]$ efficiently as the amplitude of quantum state of only $O(D)$ qubits. Roughly speaking, as in Theorem 2.2, these weights can be given by the discrete ridgelet transform $\mathcal{R}[f]$ of f, which is implementable efficiently by QRT. In particular, if we initially have a quantum state $|f\rangle = \sum_{\boldsymbol{x}} f(\boldsymbol{x}) |\boldsymbol{x}\rangle$, then QRT of $|f\rangle$ can prepare $\boldsymbol{R}|f\rangle = \sum_{\boldsymbol{a},b} \mathcal{R}[f](\boldsymbol{a},b) |\boldsymbol{a},b\rangle$ in time $\widetilde{O}(D)$ as shown in Theorem 3.1, where $\widetilde{O}$ may ignore polylogarithmic factors. A measurement of this quantum state $\boldsymbol{R}|f\rangle$ in basis $\{|\boldsymbol{a},b\rangle\}$ provides a measurement outcome $(\boldsymbol{a},b)$ sampled from a probability distribution proportional to the square of the amplitude, i.e., $|\mathcal{R}[f](\boldsymbol{a},b)|^2$. In this way, we can find parameter $(\boldsymbol{a},b)$ for a node with large $|\mathcal{R}[f](\boldsymbol{a},b)|^2$ with high probability, in runtime $\widetilde{O}(D)$ per sampling. The state is corrupted by the measurement, and to perform the sampling N times, we repeat the preparation and measurement N times. To sample $(\boldsymbol{a},b)$ for all high-weight nodes with high probability in a theoretically guaranteed way, for $\Delta > 0$, we introduce an **optimized probability distribution**

$$p_{\lambda,\Delta}^*(\boldsymbol{a},b) := \frac{1}{Z}\frac{|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)|^2}{|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)|^2 + \Delta}, \tag{23}$$

where Z is a constant for normalization $\sum_{\boldsymbol{a},b} p_{\lambda,\Delta}^*(\boldsymbol{a},b) = 1$. Appropriate Δ for our task in (21) will be specified later in Theorem 4.2. To sample from $p_{\lambda,\Delta}^*$, we prepare

$$|p_{\lambda,\Delta}^*\rangle := \frac{1}{\sqrt{Z}} \sum_{\boldsymbol{a},b} \frac{P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)}{\sqrt{|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)|^2 + \Delta}} |\boldsymbol{a},b\rangle, \tag{24}$$

followed by performing the measurement of $|p_{\lambda,\Delta}^*\rangle$ in the same way as described above.

To apply the above idea to practical tasks of machine learning, it is important to deal with a conventional situation where the data (e.g., examples of input-output pairs of f) are given by classical bit strings rather than quantum states; thus, a critical issue for our algorithm should be how to give such a quantum state from classical data. To achieve the overall task using QRT, we first need to input the classical data by converting the data into a quantum state, then apply QRT to the quantum state, and finally perform a measurement to obtain a classical output from the quantum state. We also remark that, in some other proposals of QML, some "quantum" data may be assumed to be given by quantum states, e.g., as a result of another quantum algorithm or physical process, and QRT is also potentially useful in such a quantum setting. But significantly, our algorithm here avoids such an assumption by explicitly clarifying how to prepare an input quantum state from the given classical examples in the task of finding the winning ticket of neural networks, as shown below.

In particular, we explicitly construct an input model for our quantum algorithm by quantum circuit as follows.

- As an input, our algorithm uses quantum circuits to prepare $|\hat{p}_{\mathrm{data}}\rangle = \sum_{\boldsymbol{x}} \sqrt{\hat{p}_{\mathrm{data}}(\boldsymbol{x})} |\boldsymbol{x}\rangle$ and $|\psi_{\mathrm{in}}\rangle \propto \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x}) |\boldsymbol{x}\rangle$. Regarding $|\hat{p}_{\mathrm{data}}\rangle$, if we prepare $|\hat{p}_{\mathrm{data}}\rangle$ and measure it in basis $\{|\boldsymbol{x}\rangle\}$, we can randomly sample $\boldsymbol{x}$ according to the empirical distribution $\hat{p}_{\mathrm{data}}(\boldsymbol{x})$. For a classical algorithm, sampling from $\hat{p}_{\mathrm{data}}$ can be easily realized in time $O(D\,\mathrm{polylog}(M))$ over M examples of D-dimensional input, by sampling $m \in \{1,\ldots,M\}$ from the uniform distribution over $O(\log(M))$ bits and outputting $\boldsymbol{x}_m \in \mathbb{Z}_P^D$ out of $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M$ stored in random access memory (RAM). As for the quantum algorithm, the preparation of $|\hat{p}_{\mathrm{data}}\rangle$ with maintaining quantum superposition may be more technical. But we show that this preparation is also implementable in time $O(D\,\mathrm{polylog}(M))$, by storing the M examples upon collecting them in a sparse binary-tree data structure (Kerenidis & Prakash, 2017) with quantum RAM (QRAM) (Giovannetti et al., 2008a;b), where QRAM is implemented explicitly as a parallelized quantum circuit of depth $O(\mathrm{polylog}(M))$ (Matteo et al., 2020; Hann et al., 2021). Using the same data structure, we also show that the preparation of $|\psi_{\mathrm{in}}\rangle$ is implemented within the same runtime $O(D\,\mathrm{polylog}(M))$. As a whole, our assumption is to store the M examples in these data structures upon collecting them, so that each preparation of $|\hat{p}_{\mathrm{data}}\rangle$ and $|\psi_{\mathrm{in}}\rangle$ has runtime $O(D\,\mathrm{polylog}(M)) = \widetilde{O}(D)$. See Appendix D for detail.

The following theorem shows that we have a quantum algorithm that can prepare and measure $|p_{\lambda,\Delta}^*\rangle$ to sample from $p_{\lambda,\Delta}^*$ within a linear runtime $\widetilde{O}(\frac{D}{\lambda\Delta})$. Our algorithm is based on the analytical formula for the solution of ridge regression in (18); in particular, with $r = g$, the formula leads to

$$|p_{\lambda,\Delta}^*\rangle \propto \left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-\frac{1}{2}} \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}\right)^{-1} \boldsymbol{R}|\psi_{\mathrm{in}}\rangle, \tag{25}$$

where $\boldsymbol{W}_\lambda := \gamma^{-1}\sum_{\boldsymbol{a},b} |P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)|^2 |\boldsymbol{a},b\rangle\langle\boldsymbol{a},b|$ and $\hat{\boldsymbol{P}}_{\mathrm{data}} := \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) |\boldsymbol{x}\rangle\langle\boldsymbol{x}|$. The algorithm is also explicitly presented as Algorithm 2 in Appendix D. See Appendix D for proof of its runtime as well.

Theorem 4.1 (Runtime of quantum algorithm for sampling from optimized probability distribution)**.** *Given $\lambda, \Delta > 0$, for any D, a quantum algorithm can prepare and measure $|p^*_{\lambda,\Delta}\rangle$ to sample from $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ within runtime $\widetilde{O}\left(\frac{D}{\lambda\Delta} \times \gamma\right)$ per sampling, where γ is a constant in (20).*

Remarkably, our construction of the quantum algorithm for Theorem 4.1 is based on two significant technical contributions. First, estimation of classical description of $|p^*_{\lambda,\Delta}\rangle$ would need $\exp(O(D))$ runtime and may cancel out the advantage of QML (Aaronson, 2015), but we avoid such slowdown. In particular, our quantum algorithm prepares $|p^*_{\lambda,\Delta}\rangle$ directly from the M examples and then measure it to obtain parameter $(\boldsymbol{a}, b)$ for a high-weight node of $\mathcal{S}[w^*_\lambda]$ per single preparation and measurement. In this way, we circumvent the costly process of expectation-value estimation throughout our algorithm. Second, we develop a technique for implementing the inverses of $\exp(O(D)) \times \exp(O(D))$ matrices in (25) with quantum computation efficiently, yet without imposing restrictive assumptions. In particular, the inverses of $\exp(O(D)) \times \exp(O(D))$ matrices are hard to compute in classical computation, and conventional techniques in QML have required sparsity or low-rankness of the matrices to implement matrix inversion with large quantum speedups (Gilyén et al., 2019). More recent quantum-inspired classical algorithms also require the low-rank assumption (Tang, 2019). However, the matrices to be inverted in our algorithm are not necessarily sparse or low-rank, and thus imposing such assumptions would limit the applicability of QML. By contrast, we avoid imposing these assumptions by directly clarifying the quantum circuits for implementing these matrices efficiently with QRT. In the existing research, this type of technique for avoiding the sparsity and low-rankness assumptions in QML was established only for Fourier transform (Yamasaki et al., 2020; Yamasaki & Sonoda, 2021). Our development discovers wide applicability of such techniques even to a broader class of transforms including ridgelet transform.

4.3. Quantum Algorithm for Finding Winning Ticket of Neural Networks and Performance Analysis

Using the quantum algorithm for sampling from $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ in Theorem 4.1, we describe **an algorithm for finding a winning ticket**, i.e., a sparse trainable subnetwork of the large original network $\mathcal{S}[w^*_\lambda]$ in (17) for approximating f. We also analyze its performance with theoretical guarantee.

We here describe our algorithm for finding a winning ticket, which is also explicitly presented as Algorithm 3 in Appendix E. In our algorithm, we repeat the sampling from $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ in total N times by the quantum algorithm in Theorem 4.1, where N is given later in (28). Letting $\hat{\mathbb{W}}$ denote the set of sampled parameters in these N repetitions, we approximate $\mathcal{S}[w^*_\lambda]$ by the subnetwork

$$\mathcal{S}[w^*_\lambda] \approx \hat{f}(\boldsymbol{x}) = \sum_{(\boldsymbol{a},b)\in\hat{\mathbb{W}}} \hat{w}^*(\boldsymbol{a}, b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P), \quad (26)$$

where we write $\hat{\boldsymbol{w}}^* := (\hat{w}^*(\boldsymbol{a}, b) \in \mathbb{R} : (\boldsymbol{a}, b) \in \hat{\mathbb{W}})$. Each sampling provides parameter $(\boldsymbol{a}, b) \in \hat{\mathbb{W}}$ of each node in the hidden layer of this subnetwork but not the value of $\hat{w}^*(\boldsymbol{a}, b)$. Once we fix $\{g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) : (\boldsymbol{a}, b) \in \hat{\mathbb{W}}\}$ of the subnetwork, we then train $\hat{\boldsymbol{w}}^*$ efficiently by the established classical algorithms for convex optimization such as stochastic gradient descent (SGD) (Harvey et al., 2019), using the M examples. Thus, the quantum speedup is not cancelled out throughout the learning including the training of $\hat{\boldsymbol{w}}^*$. In this way, we achieve our task (21) with trainability.

The following theorem guarantees Δ and N required for achieving our task (21). By combining Theorems 4.1 and 4.2 with (22) in our setting, the overall runtime of our algorithm is

$$\widetilde{O}\left(N \times \frac{D}{\lambda\Delta}\gamma\right) = \widetilde{O}\left(\frac{D}{\lambda\epsilon^{1+2/\beta}}\right) = \widetilde{O}\left(D \times \mathrm{poly}\left(\frac{1}{\epsilon}\right)\right), \quad (27)$$

dominated by the N repetitions of the sampling in Theorem 4.1. A comparison with classical algorithms analogous to our sampling-based approach is made in Sec. 4.4. See Appendix E for proof.

Theorem 4.2 (Bounds for finding winning ticket of neural networks)**.** *Given $\epsilon, \delta > 0$, there exist Δ and N satisfying*

$$\Delta = \Omega\left(\epsilon^{1+\frac{1}{\beta}}\right), \ N = O\left(\epsilon^{-\frac{1}{2\beta}} \log\left(\epsilon^{-1}\delta^{-1}\right)\right), \quad (28)$$

such that the algorithm described above returns a subnetwork $\hat{f}$ of the neural network $\mathcal{S}[w^*_\lambda]$ with the number of nodes in the hidden layer of $\hat{f}$ smaller than N, and $\hat{f}$ achieves the task of approximating $\mathcal{S}[w^*_\lambda]$ to $O(\epsilon)$ in (21) with high probability greater than $1 - \delta$.

4.4. Advantage of Using Quantum Ridgelet Transform

We numerically demonstrate **the advantage of our algorithm for winning the lottery ticket** of the neural network in Theorem 4.2. For fair comparison between our algorithm and a similar approach for classical algorithms, recall that the essential idea of our algorithm is to avoid computationally hard optimization of the initial neural network by sampling the nodes in its hidden layer to decide the basis functions $\{g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) : (\boldsymbol{a}, b) \in \hat{\mathbb{W}}\}$ in (26), followed by efficiently training their coefficients $\hat{\boldsymbol{w}}^*$ via convex optimization. This idea can be regarded as a generalized form of random features by Rahimi & Recht (2008; 2009), where one randomly samples feature maps, i.e., $\{g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P)\}$, and then find their coefficients

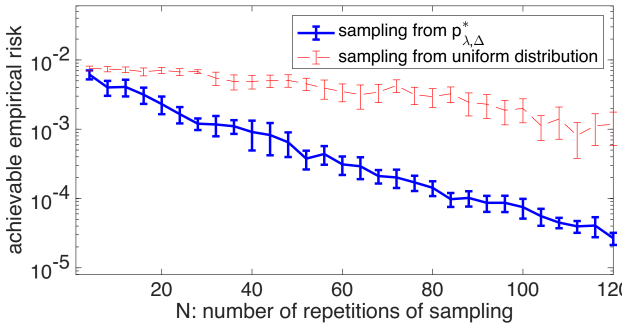

Figure 1. The empirical risks achievable with the subnetworks of the large original neural network found by N repetitions of sampling from the optimized distribution $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ in (23) via our algorithm in Theorem 4.2 (blue thick line), and that from the uniform distribution via random features (red dashed line). Each line represents the average over 20 executions of the algorithms, while each error bar represents the unbiased estimation of the standard deviation for these executions. The advantage of using our algorithm over the simple application of the random features can be order of magnitude in terms of the empirical risk in this regime.

by convex optimization. The difference is that we use the optimized distribution $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ depending on f and $\hat{p}_{\mathrm{data}}$, but the random features conventionally performs sampling from a distribution independent of f and $\hat{p}_{\mathrm{data}}$, e.g., a uniform distribution $p_{\mathrm{uniform}}(\boldsymbol{a}, b) \coloneqq \frac{1}{P^{D+1}}$. Note that a more recent work by Bach (2017) has proposed to sample optimized random features depending on the data distribution (but still not on f itself), and this sampling is also efficiently achievable with a quantum algorithm shown by Yamasaki et al. (2020); Yamasaki & Sonoda (2021); however, without QRT in this work, it would not be straightforward to apply such techniques to neural networks. Despite this difference, a quantitative advantage of our algorithm over the random features would be still unclear without numerical simulation, due to the non-orthogonality of the basis functions $\{g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P)\}$ used for the neural network.

We numerically show that our algorithm can find a subnetwork achieving a significantly better empirical risk than that obtained from the random features, as illustrated in Fig. 1. In our numerical experiment, choosing $D = 1$ and $P = 127$, we set the function f to be learned as a sine function, the empirical distribution as the uniform distribution $\hat{p}_{\mathrm{data}}(\boldsymbol{x}) = \frac{1}{P^D}$, and the activation function g and the ridgelet function r as ReLU. The sampling from $p^*_{\lambda,\Delta}$ was classically simulated via rejection sampling, and convex optimization of $\hat{w}^*$ was solved by MOSEK (ApS, 2022) and YALMIP (Löfberg, 2004). For $N \leqq 120$, we plotted the achievable empirical risk with N repetitions of the sampling from $p^*_{\lambda,\Delta}$ and that from p_{uniform}. The advantage of our algorithm in finding a sparse trainable subnetwork over the random features can be order of magnitude in terms of the empirical risk achievable by the subnetwork. We also note that a similar advantage can also be obtained in the case where we choose the activation function as a sigmoid function (tanh) rather than ReLU, which supports our results further. See Appendix F for detail.

We emphasize that our classical simulation using rejection sampling of $p^*_{\lambda,\Delta}$ is not scalable as D increases; by contrast, our results make it possible to take the advantage in higher dimension $D \gg 1$ if we can use quantum computation for accelerating the task. Our main contributions are the development of algorithmic techniques for QML and the derivation of the bounds on the performances of our algorithms with theoretical guarantees, which are not heuristic. We may be able to perform the numerical experiment only for small D at the moment because the numerical experiment for higher dimensions is computationally hard as long as we use classical computation to simulate quantum computation. However, together with our theoretical analysis, the overall results lay a solid foundation for developing fault-tolerant quantum computers to demonstrate the advantage of our algorithms for the higher dimensions in quantum experiments in the future.

5. Conclusion

We have formulated the discrete ridgelet transform that can be characterized via Fourier slice theorem and can represent any function exactly in the discretized domain. Furthermore, as a fundamental subroutine for quantum machine learning (QML), we have constructed quantum ridgelet transform (QRT), a quantum algorithm for applying D-dimensional discrete ridgelet transform to a quantum state efficiently in linear time $\widetilde{O}(D)$. We have also clarified an application of QRT for finding a sparse trainable subnetwork of a large-scale neural network to approximate a function to be learned, opening an efficient way to demonstrate lottery ticket hypothesis. These results discover a promising use of QML to accelerate the tasks for classical neural networks. Also from a broader perspective, our quantum algorithms may need a fault-tolerant quantum computer that is actively under development, and our achievement lays a solid theoretical foundation for further hardware development and social implementation toward realizing quantum computation.

Acknowledgements

Hayata Yamasaki was supported by JSPS Overseas Research Fellowship, JST PRESTO Grant Number JPMJPR201A and MEXT Quantum Leap Flagship Program (MEXT QLEAP) JPMXS0118069605, JPMXS0120351339. Sathyawageeswar Subramanian was supported by Royal Commission for the Exhibition of 1851 Research Fellowship in Science and Engineering. Sho Sonoda was supported by JST PRESTO Grant Number JPMJPR2125.

References

Aaronson, S. Read the fine print. *Nature Physics*, 11(4): 291, 2015. URL https://www.nature.com/articles/nphys3272.

Ambainis, A. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In Dürr, C. and Wilke, T. (eds.), *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 636–647, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-35-4. doi: 10.4230/LIPIcs.STACS.2012.636. URL http://drops.dagstuhl.de/opus/volltexte/2012/3426.

ApS, M. *The MOSEK optimization toolbox for MATLAB manual. Version 9.3.21.*, 2022. URL http://docs.mosek.com/9.3/toolbox/index.html.

Argüello, F. Quantum wavelet transforms of any order. *Quantum Info. Comput.*, 9(5):414–422, may 2009. ISSN 1533-7146. URL https://dl.acm.org/doi/abs/10.5555/2011791.2011796.

Bach, F. 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.

Bach, F. *Learning Theory from First Principles*. 2021. URL https://www.di.ens.fr/%7Efbach/ltfp_book.pdf.

Barron, A. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993. doi: 10.1109/18.256500. URL https://ieeexplore.ieee.org/document/256500/.

Bernstein, E. and Vazirani, U. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. doi: 10.1137/S0097539796300921. URL https://epubs.siam.org/doi/10.1137/S0097539796300921.

Beylkin, G. Discrete radon transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(2):162–172, 1987. doi: 10.1109/TASSP.1987.1165108. URL https://ieeexplore.ieee.org/document/1165108.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195, 2017. URL https://www.nature.com/articles/nature23474.

Boag, A., Bresler, Y., and Michielssen, E. A multilevel domain decomposition algorithm for fast o(n/sup 2/logn) reprojection of tomographic images. *IEEE Transactions on Image Processing*, 9(9):1573–1582, 2000. doi: 10.1109/83.862638. URL https://ieeexplore.ieee.org/document/862638.

Brady, M. L. A fast discrete approximation algorithm for the radon transform. *SIAM Journal on Computing*, 27(1): 107–119, 1998. doi: 10.1137/S0097539793256673. URL https://epubs.siam.org/doi/10.1137/S0097539793256673.

Brandt, A., Mann, J., Brodski, M., and Galun, M. A fast and accurate multilevel inversion of the radon transform. *SIAM Journal on Applied Mathematics*, 60(2):437–462, 2000. doi: 10.1137/S003613999732425X. URL https://epubs.siam.org/doi/10.1137/S003613999732425X.

Candes, E. J. *Ridgelets : theory and applications*. PhD thesis, Stanford University, 1998. URL https://searchworks.stanford.edu/view/9949708.

Carre, P. and Andres, E. Discrete analytical ridgelet transform. *Signal Processing*, 84(11):2165–2173, 2004. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2004.07.009. URL https://www.sciencedirect.com/science/article/pii/S0165168404001689. Special Section Signal Processing in Communications.

Chakraborty, S., Gilyén, A., and Jeffery, S. The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation. In Baier, C., Chatzigiannakis, I., Flocchini, P., and Leonardi, S. (eds.), *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 33:1–33:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-109-2. doi: 10.4230/LIPIcs.ICALP.2019.33. URL http://drops.dagstuhl.de/opus/volltexte/2019/10609.

Chen, S., Cotler, J., Huang, H., and Li, J. Exponential separations between learning with and without quantum memory. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 574–585, Los Alamitos, CA, USA, feb 2022a. IEEE Computer Society. doi: 10.1109/FOCS52979.2021.00063. URL https://doi.ieeecomputersociety.org/10.1109/FOCS52979.2021.00063.

Chen, X., Zhang, J., and Wang, Z. Peek-a-boo: What (more) is disguised in a randomly weighted neural network, and how to find it efficiently. In *International Conference on*

Learning Representations, 2022b. URL https://openreview.net/forum?id=moHCzz6D5H3.

Childs, A. M., Kothari, R., and Somma, R. D. 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://epubs.siam.org/doi/10.1137/16M1087072.

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., and Wossnig, L. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018. doi: 10.1098/rspa.2017.0551. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2017.0551.

Cleve, R. and Watrous, J. Fast parallel circuits for the quantum fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 526–536, 2000. doi: 10.1109/SFCS.2000.892140. URL https://ieeexplore.ieee.org/document/892140.

Coppersmith, D. An approximate fourier transform useful in quantum factoring. *IBM Research Report*, pp. RC–19642, 1994. URL https://dominoweb.draco.res.ibm.com/reports/8472.ps.gz.

de Wolf, R. Quantum computing: Lecture notes, 2019. URL https://arxiv.org/abs/1907.09415.

Do, M. and Vetterli, M. The finite ridgelet transform for image representation. *IEEE Transactions on Image Processing*, 12(1):16–28, 2003. doi: 10.1109/TIP.2002.806252. URL https://ieeexplore.ieee.org/document/1187351.

Donoho, D. L. Unconditional bases are optimal bases for data compression and for statistical estimation. *Applied and Computational Harmonic Analysis*, 1(1):100–115, 1993. ISSN 1063-5203. doi: https://doi.org/10.1006/acha.1993.1008. URL https://www.sciencedirect.com/science/article/pii/S1063520383710080.

Fadili, J. and Starck, J.-L. *Curvelets and Ridgelets*, pp. 754–773. Springer New York, New York, NY, 2012. ISBN 978-1-4614-1800-9. doi: 10.1007/978-1-4614-1800-9_48. URL https://link.springer.com/referenceworkentry/10.1007/978-1-4614-1800-9_48.

Fijany, A. and Williams, C. P. Quantum wavelet transforms: Fast algorithms and complete circuits. In *Selected Papers from the First NASA International Conference on Quantum Computing and Quantum Communications*, QCQC '98, pp. 10–33, Berlin, Heidelberg, 1998. Springer-Verlag. ISBN 354065514X. URL https://dl.acm.org/doi/abs/10.5555/645812.670803.

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

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ig-VyQc-MLK.

Gilyén, A., Su, Y., Low, G. H., and Wiebe, N. 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, pp. 193–204, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367059. doi: 10.1145/3313276.3316366. URL https://dl.acm.org/doi/10.1145/3313276.3316366.

Giovannetti, V., Lloyd, S., and Maccone, L. Architectures for a quantum random access memory. *Phys. Rev. A*, 78:052310, Nov 2008a. doi: 10.1103/PhysRevA.78.052310. URL https://link.aps.org/doi/10.1103/PhysRevA.78.052310.

Giovannetti, V., Lloyd, S., and Maccone, L. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, Apr 2008b. doi: 10.1103/PhysRevLett.100.160501. URL https://link.aps.org/doi/10.1103/PhysRevLett.100.160501.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org.

Grover, L. and Rudolph, T. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. URL https://arxiv.org/abs/quant-ph/0208112.

Götz, W. and Druckmüller, H. A fast digital radon transform—an efficient means for evaluating the hough transform. *Pattern Recognition*, 29(4):711–718, 1996. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(96)00015-5. URL https://www.sciencedirect.com/science/article/pii/0031320396000155.

Hales, L. and Hallgren, S. An improved quantum fourier transform algorithm and applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 515–525, 2000. doi: 10.1109/SFCS.2000.892139. URL https://ieeexplore.ieee.org/document/892139.

Hann, C. T., Lee, G., Girvin, S., and Jiang, L. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2:020311, Apr 2021. doi: 10.1103/PRXQuantum.2.020311. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.020311.

Harvey, N. J. A., Liaw, C., Plan, Y., and Randhawa, S. Tight analyses for non-smooth stochastic gradient descent. In Beygelzimer, A. and Hsu, D. (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1579–1613. PMLR, 25–28 Jun 2019. URL https://proceedings.mlr.press/v99/harvey19a.html.

Hayakawa, S. and Suzuki, T. On the minimax optimality and superiority of deep neural network learning over sparse parameter spaces. *Neural Networks*, 123:343–361, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.12.014. URL https://www.sciencedirect.com/science/article/pii/S089360801930406X.

Helbert, D., Carre, P., and Andres, E. 3-d discrete analytical ridgelet transform. *IEEE Transactions on Image Processing*, 15(12):3701–3714, 2006. doi: 10.1109/TIP.2006.881936. URL https://ieeexplore.ieee.org/abstract/document/4011958.

Hoyer, P. Efficient quantum transforms, 1997. URL https://arxiv.org/abs/quant-ph/9702028.

Huang, H.-Y., Kueng, R., and Preskill, J. Information-theoretic bounds on quantum advantage in machine learning. *Phys. Rev. Lett.*, 126:190505, May 2021. doi: 10.1103/PhysRevLett.126.190505. URL https://link.aps.org/doi/10.1103/PhysRevLett.126.190505.

Huang, H.-Y., Broughton, M., Cotler, J., Chen, S., Li, J., Mohseni, M., Neven, H., Babbush, R., Kueng, R., Preskill, J., and McClean, J. R. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 2022. doi: 10.1126/science.abn7293. URL https://www.science.org/doi/abs/10.1126/science.abn7293.

Kelley, B. and Madisetti, V. The fast discrete radon transform. i. theory. *IEEE Transactions on Image Processing*, 2(3):382–400, 1993. doi: 10.1109/83.236530. URL https://ieeexplore.ieee.org/document/236530.

Kerenidis, I. and Prakash, A. Quantum Recommendation Systems. In Papadimitriou, C. H. (ed.), *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 49:1–49:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-029-3. doi: 10.4230/LIPIcs.ITCS.2017.49. URL http://drops.dagstuhl.de/opus/volltexte/2017/8154.

Kitaev, A. Y. Quantum measurements and the abelian stabilizer problem, 1995. URL https://arxiv.org/abs/quant-ph/9511026.

Labunets, V., Labunets-Rundblad, E., and Astola, J. Fast classical and quantum fractional haar wavelet transforms. In *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.*, pp. 564–569, 2001a. doi: 10.1109/ISPA.2001.938692. URL https://ieeexplore.ieee.org/abstract/document/938692.

Labunets, V., Labunets-Rundblad, E., Egiazarian, K., and Astola, J. Fast classical and quantum fractional walsh transforms. In *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.*, pp. 558–563, 2001b. doi: 10.1109/ISPA.2001.938691. URL https://ieeexplore.ieee.org/abstract/document/938691.

Lee, N., Ajanthan, T., and Torr, P. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

Li, H.-S., Fan, P., ying Xia, H., and Song, S. Quantum multi-level wavelet transforms. *Information Sciences*, 504:113–135, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.07.057. URL https://www.sciencedirect.com/science/article/pii/S0020025519306632.

Li, H.-S., Fan, P., Peng, H., Song, S., and Long, G.-L. Multilevel 2-d quantum wavelet transforms. *IEEE Transactions on Cybernetics*, 52(8):8467–8480, 2022. doi: 10.1109/TCYB.2021.3049509. URL https://ieeexplore.ieee.org/abstract/document/9337176.

Liu, Y., Arunachalam, S., and Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17:1013, 2021. URL https://www.nature.com/articles/s41567-021-01287-z.

Liu, Y.-K. Quantum algorithms using the curvelet transform. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pp. 391–400, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585062. doi: 10.1145/1536414.1536469. URL https://dl.acm.org/doi/10.1145/1536414.1536469.

Löfberg, J. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. URL https://yalmip.github.io/.

Ma, G., Li, H., and Zhao, J. Quantum radon transforms and their applications. *IEEE Transactions on Quantum Engineering*, 3:1–16, 2022. doi: 10.1109/TQE.2021.3134648. URL https://ieeexplore.ieee.org/abstract/document/9648027.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6682–6691. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/malach20a.html.

Matteo, O. D., Gheorghiu, V., and Mosca, M. Fault-tolerant resource estimation of quantum random-access memories. *IEEE Transactions on Quantum Engineering*, 1:1–13, 2020. doi: 10.1109/TQE.2020.2965803. URL https://ieeexplore.ieee.org/document/8962352.

Mosca, M. and Zalka, C. Exact quantum fourier transforms and discrete logarithm algorithms. *International Journal of Quantum Information*, 02(01):91–100, 2004. doi: 10.1142/S0219749904000109. URL https://www.worldscientific.com/doi/abs/10.1142/S0219749904000109.

Murata, N. 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.

Nielsen, M. A. and Chuang, I. L. *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.

Orseau, L., Hutter, M., and Rivasplata, O. Logarithmic pruning is all you need. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2925–2934. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1e9491470749d5b0e361ce4f0b24d037-Paper.pdf.

Pensia, A., Rajput, S., Nagle, A., Vishwakarma, H., and Papailiopoulos, D. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2599–2610. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1b742ae215adf18b75449c6e272fd92d-Paper.pdf.

Press, W. H. Discrete radon transform has an exact, fast inverse and generalizes to operations other than sums along lines. *Proceedings of the National Academy of Sciences*, 103(51):19249–19254, 2006. doi: 10.1073/pnas.0609228103. URL https://www.pnas.org/doi/abs/10.1073/pnas.0609228103.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T. (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1177–1184. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1313–1320. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3495-weighted-sums-of-random-kitchen-sinks-replacing-minimization-with-randomization-in-learning.pdf.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What' s hidden in a randomly weighted neural network? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11890–11899, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.011

91. URL https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01191.

Rubin, B. The calderón reproducing formula, windowed x-ray transforms, and radon transforms in l...-spaces. *The journal of Fourier analysis and applications [[Elektronische Ressource]]*, 4(2):175–198, 1998. URL https://link.springer.com/article/10.1007/BF02475988.

Schuld, M. and Petruccione, F. *Machine learning with quantum computers*. Springer, 2021. URL https://link.springer.com/book/10.1007/978-3-030-83098-4.

Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi: 10.1137/S0097539795293172. URL https://epubs.siam.org/doi/10.1137/S0097539795293172.

Simon, D. On the power of quantum computation. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 116–123, 1994. doi: 10.1109/SFCS.1994.365701. URL https://ieeexplore.ieee.org/document/365701.

Sonoda, S. and Murata, N. Sampling hidden parameters from oracle distribution. In Wermter, S., Weber, C., Duch, W., Honkela, T., Koprinkova-Hristova, P., Magg, S., Palm, G., and Villa, A. E. P. (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2014*, pp. 539–546, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11179-7. URL https://link.springer.com/chapter/10.1007/978-3-319-11179-7_68.

Sonoda, S. and Murata, N. 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.

Sonoda, S., Ishikawa, I., and Ikeda, M. Ridge regression with over-parametrized two-layer networks converge to ridgelet spectrum. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 2674–2682. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/sonoda21a.html.

Sonoda, S., Ishikawa, I., and Ikeda, M. Fully-connected network on noncompact symmetric space and ridgelet transform based on helgason-Fourier analysis. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20405–20422. PMLR, 17–23 Jul 2022a. URL https://proceedings.mlr.press/v162/sonoda22a.html.

Sonoda, S., Ishikawa, I., and Ikeda, M. Universality of group convolutional neural networks based on ridgelet analysis on groups. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=ebCk2FNI1za.

Starck, J.-L., Murtagh, F., and Fadili, J. M. *The Ridgelet and Curvelet Transforms*, pp. 89–118. Cambridge University Press, 2010. doi: 10.1017/CBO9780511730344.006. URL https://www.cambridge.org/core/books/abs/sparse-image-and-signal-processing/ridgelet-and-curvelet-transforms/D84CAD0CEB84E2940B43B4ED298C0C62.

Sweke, R., Seifert, J.-P., Hangleiter, D., and Eisert, J. On the Quantum versus Classical Learnability of Discrete Distributions. *Quantum*, 5:417, March 2021. ISSN 2521-327X. doi: 10.22331/q-2021-03-23-417. URL https://quantum-journal.org/papers/q-2021-03-23-417/.

Taha, S. M. R. *Wavelets and Multiwavelets Implementation Using Quantum Computing*, pp. 153–170. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23479-3. doi: 10.1007/978-3-319-23479-3_7. URL https://link.springer.com/chapter/10.1007/978-3-319-23479-3_7.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6377–6389. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf.

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

Tseng, C.-C. and Hwang, T.-M. Quantum circuit design of 8 /spl times/ 8 discrete hartley transform. In *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 3, pp. III–397, 2004. doi: 10.1109/ISCAS.2004.1328767. URL https://ieeexplore.ieee.org/abstract/document/1328767.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=SkgsACVKPH.

Wang, Y., Zhang, X., Xie, L., Zhou, J., Su, H., Zhang, B., and Hu, X. Pruning from scratch. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12273–12280, Apr. 2020b. doi: 10.1609/aaai.v34i07.6910. URL https://ojs.aaai.org/index.php/AAAI/article/view/6910.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15173–15184. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ad1f8bb9b51f023cdc80cf94bb615aa9-Paper.pdf.

Yamakawa, T. and Zhandry, M. Verifiable quantum advantage without structure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 69–74, Los Alamitos, CA, USA, nov 2022. IEEE Computer Society. doi: 10.1109/FOCS54457.2022.00014. URL https://doi.ieeecomputersociety.org/10.1109/FOCS54457.2022.00014.

Yamasaki, H. and Sonoda, S. Exponential error convergence in data classification with optimized random features: Acceleration by quantum machine learning, 2021. URL https://arxiv.org/abs/2106.09028.

Yamasaki, H., Subramanian, S., Sonoda, S., and Koashi, M. Learning with optimized random features: Exponential speedup by quantum machine learning without sparsity and low-rank assumptions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13674–13687. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/9ddb9dd5d8aee9a76bf217a2a3c54833-Paper.pdf.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf.

Appendices

Appendices are organized as follows. In Appendix A, we provide a proof of Theorem 2.1 in Sec. 2.1. In Appendix B, we provide a proof of Theorem 2.2 in Sec. 2.2. In Appendix C, we summarize basic notions of quantum computation required for the analysis and then provide a proof of Theorem 3.1 in Sec. 3. In Appendix D, we clarify the explicit implementation and the runtime of the input model for our quantum algorithm for Theorem 4.1 in Sec. 4.2 and then provide the proof of Theorem 4.1. In Appendix E, we provide the proof of Theorem 4.2 in Sec. 4.3. In Appendix F, we describe the detail of the parameters chosen for the numerical experiment in Sec. 4.4.

A. Proof on Fourier Slice Theorem for Discrete Ridgelet Transform

We provide a proof of Theorem 2.1 in Sec. 2.1.

Proof of Theorem 2.1. It holds that

$$\begin{aligned}
&\mathcal{R}[f](\boldsymbol{a}, b) &(29)\\
&= P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) r((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) &(30)\\
&= P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) \overline{r((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P)} &(31)\\
&= P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x})\, \overline{P^{-\frac{1}{2}} \sum_{v} \mathcal{F}_1[r](v) \mathrm{e}^{\frac{2\pi \mathrm{i} v(\boldsymbol{a}^\top \boldsymbol{x} - b)}{P}}} &(32)\\
&= P^{-\frac{1}{2}} \sum_{v \in \mathbb{Z}_P} \left(P^{-\frac{D}{2}} \sum_{\boldsymbol{x}} f(\boldsymbol{x}) \mathrm{e}^{\frac{-2\pi \mathrm{i} v \boldsymbol{a}^\top \boldsymbol{x}}{P}}\right) \overline{\mathcal{F}_1[r](v)} \mathrm{e}^{\frac{2\pi \mathrm{i} v b}{P}} &(33)\\
&= P^{-\frac{1}{2}} \sum_{v \in \mathbb{Z}_P} \mathcal{F}_D[f](v\boldsymbol{a} \bmod P)\, \overline{\mathcal{F}_1[r](v)}\, \mathrm{e}^{\frac{2\pi \mathrm{i} v b}{P}}. &(34)
\end{aligned}$$

Therefore, the discrete Fourier transform for b leads to

$$\mathcal{F}_1[\mathcal{R}[f](\boldsymbol{a}, \cdot)] = \mathcal{F}_D[f](v\boldsymbol{a} \bmod P)\, \overline{\mathcal{F}_1[r](v)}, \tag{35}$$

which yields the conclusion. □

B. Proof on Exact Representation of Function by Discretized Neural Network

We provide a proof of Theorem 2.2 in Sec. 2.2.

Proof of Theorem 2.2. In this proof, we write $w = \mathcal{R}[f]$ for simplicity of notation. By (7), we have

$$\begin{aligned}
\mathcal{S}[w](\boldsymbol{x}) &= P^{-\frac{D}{2}} \sum_{\boldsymbol{a}} [w(\boldsymbol{a}, \cdot) * g(\cdot)](\boldsymbol{a}^\top \boldsymbol{x}) &(36)\\
&= P^{-\frac{D}{2}} \sum_{\boldsymbol{a}} \sum_{v} \mathcal{F}_1[w(\boldsymbol{a}, \cdot)](v)\, \mathcal{F}_1[g](v)\, \mathrm{e}^{\frac{2\pi \mathrm{i} v \boldsymbol{a}^\top \boldsymbol{x}}{P}}, &(37)
\end{aligned}$$

where we use

$$f * g(y) := \sum_{b} f(b) g(y - b) = \sum_{v} \mathcal{F}_1[f](v) \mathcal{F}_1[g](v) \mathrm{e}^{\frac{2\pi \mathrm{i} v y}{P}}. \tag{38}$$

Then, applying Theorem 2.1 to $w = \mathcal{R}[f]$, we have

$$(37) = P^{-\frac{D}{2}} \sum_{\boldsymbol{a}} \sum_{v} \mathcal{F}_D[f](v\boldsymbol{a} \bmod P)\, \overline{\mathcal{F}_1[r](v)}\, \mathcal{F}_1[g](v)\, \mathrm{e}^{\frac{2\pi \mathrm{i} v \boldsymbol{a}^\top \boldsymbol{x}}{G}}. \tag{39}$$

To evaluate the right-hand side, we use the fact that P is a prime. In this case, for any $v \neq 0$, the set $\mathbb{Z}_P = \{0, 1, \ldots, P-1\}$ is identical to $\{0, v, \ldots, (P-1)v\}$. Thus, recalling $\mathcal{F}_1[g](0) = \sum_b g(b) = 0$ as in (5) and letting $\boldsymbol{a}' \in \mathbb{Z}_P^D$ denote the unique vector satisfying $\boldsymbol{a}' = v\boldsymbol{a} \bmod P \in \mathbb{Z}_P^D$, we obtain

$$(39) = P^{-D/2} \sum_{v \in \mathbb{Z}_P} \sum_{\boldsymbol{a}' \in \mathbb{Z}_P^D} \mathcal{F}_D[f](\boldsymbol{a}') \, \overline{\mathcal{F}_1[r](v)} \, \mathcal{F}_1[g](v) \, \mathrm{e}^{\frac{2\pi \mathrm{i} \boldsymbol{a}'^\top \boldsymbol{x}}{P}} \tag{40}$$

$$= \left(\sum_v \mathcal{F}_1[g](v) \overline{\mathcal{F}_1[r](v)} \right) \times \left(P^{-D/2} \sum_{\boldsymbol{a}'} \mathcal{F}_D[f](\boldsymbol{a}') \mathrm{e}^{\frac{2\pi \mathrm{i} \boldsymbol{a}'^\top \boldsymbol{x}}{P}} \right) \tag{41}$$

$$= C_{g,r} f(\boldsymbol{x}). \tag{42}$$

Therefore, it holds that

$$f(\boldsymbol{x}) = \frac{1}{C_{g,r}} \mathcal{S}[w](\boldsymbol{x}), \tag{43}$$

which yields the conclusion. □

C. Proof on Runtime of Quantum Ridgelet Transform

In this section, we summarize basic notions of quantum computation; for more detial, see the textbooks and the lecture notes by, e.g., Nielsen & Chuang (2011); de Wolf (2019). Then, we provide a proof of Theorem 3.1 in Sec. 3.

Analogously to a bit $\{0,1\}$ in classical computation, the unit of quantum computation is a quantum bit (qubit), mathematically represented by $\mathbb{C}^2$, i.e., a 2-dimensional complex Hilbert space. A fixed orthonormal basis of a qubit $\mathbb{C}^2$ is denoted by $\{\left|0\right\rangle \coloneqq \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right), \left|1\right\rangle \coloneqq \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)\}$. Similar to a bit taking a state $b \in \{0,1\}$, a qubit takes a quantum state $\left|\psi\right\rangle = \alpha_0 \left|0\right\rangle + \alpha_1 \left|1\right\rangle = \left(\begin{smallmatrix}\alpha_0\\\alpha_1\end{smallmatrix}\right) \in \mathbb{C}^2$. While a register of m bits takes values in $\{0,1\}^m$, a quantum register of m qubits is represented by the tensor-product space $\left(\mathbb{C}^2\right)^{\otimes m} \cong \mathbb{C}^{2^m}$, i.e., a 2^m-dimensional Hilbert space. We may use $=$ rather than $\cong$ to represent isomorphism for brevity. We let $\mathcal{H}$ denote a finite-dimensional Hilbert space representing a quantum register; that is, an m-qubit register is $\mathcal{H} = \mathbb{C}^{2^m}$. A fixed orthonormal basis $\{\left|x\right\rangle : x \in \{0, \ldots, 2^m - 1\}\}$ labeled by m-bit strings, or the corresponding integers, is called the *standard basis* of $\mathcal{H}$. A state of $\mathcal{H}$ can be denoted by $\left|\psi\right\rangle = \sum_{x=0}^{2^m-1} \alpha_x \left|x\right\rangle \in \mathcal{H}$. Note that any quantum state $\left|\psi\right\rangle$ requires an L^2 normalization condition $\left\|\left|\psi\right\rangle\right\|_2 = 1$, and for any $\theta \in \mathbb{R}$, $\left|\psi\right\rangle$ is identified with $\mathrm{e}^{\mathrm{i}\theta}\left|\psi\right\rangle$.

In the bra-ket notation, the conjugate transpose of the column vector $\left|\psi\right\rangle$ is a row vector denoted by $\left\langle\psi\right|$, where $\left\langle\psi\right|$ and $\left|\psi\right\rangle$ may be called a bra and a ket, respectively. The inner product of $\left|\psi\right\rangle$ and $\left|\phi\right\rangle$ is denoted by $\left\langle\psi \mid \phi\right\rangle$, while their outer product $\left|\psi\right\rangle\left\langle\phi\right|$ is a matrix. The conjugate transpose of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^\dagger$, and the transpose of $\boldsymbol{A}$ with respect to the standard basis is denoted by $\boldsymbol{A}^\top$.

A measurement of a quantum state $\left|\psi\right\rangle$ is a sampling process that returns a randomly chosen bit string from the quantum state. An m-qubit state $\left|\psi\right\rangle = \sum_{x=0}^{2^m-1} \alpha_x \left|x\right\rangle$ is said to be in a superposition of the basis states $\left|x\right\rangle$s. A measurement of $\left|\psi\right\rangle$ in the standard basis $\{\left|x\right\rangle\}$ provides a random m-bit integer $x \in \{0, \ldots, 2^m - 1\}$ as outcome, with probability $p(x) = \left|\alpha_x\right|^2$. After the measurement, the state changes from $\left|\psi\right\rangle$ to $\left|x\right\rangle$ corresponding to the obtained outcome x, and loses the randomness in $\left|\psi\right\rangle$; that is, to iterate the same sampling as this measurement, we need to prepare $\left|\psi\right\rangle$ repeatedly for each iteration. For two registers $\mathcal{H}^A \otimes \mathcal{H}^B$ and their state $\left|\phi\right\rangle^{AB} = \sum_{x,x} \alpha_{x,x'} \left|x\right\rangle^A \otimes \left|x'\right\rangle^B \in \mathcal{H}^A \otimes \mathcal{H}^B$, a measurement of the register $\mathcal{H}^B$ for $\left|\phi\right\rangle^{AB}$ in the standard basis $\{\left|x'\right\rangle^B\}$ of $\mathcal{H}^B$ yields an outcome x' with probability $p(x') = \sum_x p(x, x')$, where $p(x,x') = \left|\alpha_{x,x'}\right|^2$. The superscripts of a state or an operator represent which register the state or the operator belongs to, while we may omit the superscripts if it is clear from the context.

A quantum algorithm starts by initializing m qubits in a fixed state $\left|0\right\rangle^{\otimes m}$, which we may write as $\left|0\right\rangle$ if m is clear from the context. Then, we apply a 2^m-dimensional unitary operator $\boldsymbol{U}$ to $\left|0\right\rangle^{\otimes m}$, to prepare a state $\boldsymbol{U}\left|0\right\rangle^{\otimes m}$. Finally, a measurement of $\boldsymbol{U}\left|0\right\rangle^{\otimes m}$ is performed to sample an m-bit integer from a probability distribution given by $\boldsymbol{U}\left|0\right\rangle^{\otimes m}$. Analogously to classical logic-gate circuits, $\boldsymbol{U}$ is represented by a quantum circuit composed of sequential applications of unitaries acting at most two qubits at a time. Each of these unitaries is called an elementary quantum gate. The runtime of a quantum algorithm represented by a quantum circuit is determined by the number of applications of elementary quantum gates in the circuit.

Using these notions, the proof of Theorem 3.1 in Sec. 3 is shown as follows.

Algorithm 2 Quantum algorithm for sampling from optimized probability distribution for winning ticket of neural networks.

```
Require: $\lambda, \Delta > 0$, assumptions in Sec. 4.1, the input model described in Sec. 4.2.
Ensure: Parameters $(\boldsymbol{a}, b)$ sampled from the optimized probability distribution $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$ in (23).
1: Prepare a quantum state $|\psi_{\mathrm{in}}\rangle \propto \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x}) |\boldsymbol{x}\rangle$.
2: Apply QRT to obtain a state $\boldsymbol{R} |\psi_{\mathrm{in}}\rangle$.
3: Apply $(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I})^{-1}$ by quantum singular value transformation (QSVT) to obtain
   $$\frac{1}{\sqrt{\gamma}} \sum_{\boldsymbol{a},b} P^{-\frac{D}{2}} w^*_\lambda(\boldsymbol{a}, b) |\boldsymbol{a}, b\rangle \propto (\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I})^{-1} \boldsymbol{R} |\psi_{\mathrm{in}}\rangle . \quad (44)$$
4: Apply $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I})^{-\frac{1}{2}}$ by QSVT to obtain
   $$|p^*_{\lambda,\Delta}\rangle \propto \left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-\frac{1}{2}} (\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I})^{-1} \boldsymbol{R} |\psi_{\mathrm{in}}\rangle . \quad (45)$$
5: Perform a measurement in the standard basis $\{|\boldsymbol{a}, b\rangle\}$ to sample $(\boldsymbol{a}, b)$ as the outcome according to the probability distribution $p^*_{\lambda,\Delta}(\boldsymbol{a}, b)$.
6: Return $(\boldsymbol{a}, b)$.
```

Proof of Theorem 3.1. The runtime of Algorithm 1 is domianated by Step 2 as shown in the following.

Step 1 is performed within runtime $O(\mathrm{polylog}(P))$ due to our assumption that $|r\rangle = \sum_b r(b) |b\rangle$ can be prepared in time $O(\mathrm{polylog}(P))$.

Step 2 is dominated by the runtime of $\boldsymbol{F}_P^{\otimes D}$, which is $O(D \times \mathrm{polylog}(P))$ since the runtime of $\boldsymbol{F}_P$ is $O(\mathrm{polylog}(P))$ as shown in (12). The inverse $\boldsymbol{F}_P^\dagger$ has the same runtime as $\boldsymbol{F}_P$ since $\boldsymbol{F}_P^\dagger$ is implemented by applying the inverse of each gate in the quantum circuit for $\boldsymbol{F}_P$ in the reverse order. A techinical remark is that, for simplicity of presentation, we write our statement and the proof based on the exact implementation of $\boldsymbol{F}_P$ by Mosca & Zalka (2004) to avoid writing the polylogarithmic error factors that may arise in approximate implementations of $\boldsymbol{F}_P$ such as those by Kitaev (1995); Hales & Hallgren (2000). Even if one uses these approximate implementations of $\boldsymbol{F}_P$, our theorem follows from the same argument with the polylogarithmic error factors multiplied. We also note that some of the other implementations of QFT such as those by Coppersmith (1994); Cleve & Watrous (2000) are targeted at $P = 2^n$ for $n = 1, 2, \ldots$, and our algorithm does not use these implementations since P is a prime number in our setting.

Step 3 is performed within runtime $O(\mathrm{polylog}(P))$ by arithmetics on $O(\log(P))$ qubits. This is implemented by writing the classical computation of the arithmetic in a reversible way as a classical circuit and replacing each Toffoli gate in the classical circuit with the quantum Toffoli gate to obtain the quantum circuit for the arithmetics part.

Step 4 has the runtime of $O(\mathrm{polylog}(P))$ for $\boldsymbol{F}_P^\dagger$ as discussed in Step 2 as well.

Consequently, the overall runtime is dominated by that of Step 2, i.e., $O(D \times \mathrm{polylog}(P))$. □

D. Proof on Runtime of Quantum Algorithm for Sampling from Optimized Probability Distribution

In this section, we clarify the explicit implementation and the runtime of the input model for our algorithm of Theorem 4.1 in Sec. 4.2. Then, We provide the proof of Theorem 4.1. Our algorithm for Theorem 4.1 is shown in Algorithm 2. See also Appendix C for the notations on quantum computation.

As an input model, Algorithm 2 uses preparation of quantum states

$$|\hat{p}_{\mathrm{data}}\rangle \coloneqq \sum_{\boldsymbol{x}} \sqrt{\hat{p}_{\mathrm{data}}(\boldsymbol{x})} |\boldsymbol{x}\rangle , \tag{46}$$

$$|\psi_{\mathrm{in}}\rangle \coloneqq \frac{\sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x}) |\boldsymbol{x}\rangle}{\sqrt{\sum_{\boldsymbol{x}} |\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})|^2}}, \tag{47}$$

where $\hat{p}_{\mathrm{data}}$ is the empirical distribution of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ for the M input-output pairs of examples $(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_M, f(\boldsymbol{x}_M)) \in \mathbb{Z}_P^D \times \mathbb{R}$. We will explain the implementation of the input model by quantum circuit, so as to show that the runtime of the input model in terms of the circuit depth can be bounded by $O(D \operatorname{polylog}(M))$ including the runtime of quantum random access memory (QRAM) (Giovannetti et al., 2008a;b) used for the implementation. In summary, upon collecting the M examples, we will construct an $O(M)$-size sparse data structure shown by Kerenidis & Prakash (2017), and the state preparation can be efficiently conducted with the algorithm by Grover & Rudolph (2002) using this data structure combined with QRAM; in addition, it is known that QRAM used for this input model is implementable explicitly as a parallelized quantum circuit of a $O(\operatorname{polylog}(M))$ depth using at most $O(M)$ qubits as shown, e.g., by Matteo et al. (2020); Hann et al. (2021). In the following, we explain the detail of these facts to avoid any potential confusion about feasibly of the input model for Algorithm 2 in our setting.

We explain how to construct the sparse data structure shown by Kerenidis & Prakash (2017) in our setting. In the following, we describe the data structure for $|\hat{p}_{\mathrm{data}}\rangle$ in detail, and then clarify the difference between $|\hat{p}_{\mathrm{data}}\rangle$ and $|\psi_{\mathrm{in}}\rangle$. For the preparation of $|\hat{p}_{\mathrm{data}}\rangle$, along with collecting $(\boldsymbol{x}_m, f(\boldsymbol{x}_m))$ one by one for each $m \in \{1, \ldots, M\}$, we are to perform a preprocessing to count the number of examples and store the empirical distribution in a sparse data structure proposed by Kerenidis & Prakash (2017). To describe this sparse data structure, we first explain the underlying dense binary tree (which we introduce for explanation but never store in the memory), and then clarify a sparse version of the binary tree to be stored in the memory. Each leaf of the underlying dense binary tree represents a set $\{\boldsymbol{x}\}$ for each $\boldsymbol{x} \in \mathbb{Z}_P^D$; i.e., the underlying dense binary tree has $O(P^D)$ leaves. Each parent in this dense binary tree represents the sum of the two sets represented by its two children; thus, the root of the tree represents $\mathbb{Z}_P^D$. With this definition, each node in the dense binary tree aims at storing the number of examples in the set represented by the node; e.g., a leaf representing $\{\boldsymbol{x}\}$ stores the cardinality of $\{m \in \{1, \ldots, M\} : \boldsymbol{x}_m = \boldsymbol{x}\}$. As for the sparse version of this binary tree, each leaf for $\{\boldsymbol{x}\}$ counts and stores the number of examples satisfying $\boldsymbol{x}_m = \boldsymbol{x}$ in the same way, but the sparse data structure does not store leaves with zero example. Each parent in the sparse data structure stores the sum of the counts for its children in the tree in the same way, but the sparse data structure does not store branches with zero example. As a whole, the root should store the number of all the examples, i.e., M. In this sparse data structure, the number of nodes at each depth of the tree is at most M, and the height of the tree is $O(D \log(P))$. To construct this sparse data structure for $|\hat{p}_{\mathrm{data}}\rangle$, we initialize the counts in all the nodes as 0, and for each $\boldsymbol{x}_m$ of the M examples, we increment the count in the leaf for $\{\boldsymbol{x}_m\}$ and those in the corresponding ascendants of this leaf, where the increment for each example can be performed within poly-logarithmic time $O(\operatorname{polylog}(M))$ as shown by Kerenidis & Prakash (2017). Therefore, the runtime of this preprocessing for all M examples is $\widetilde{O}(M)$, where $\widetilde{O}$ ignores the poly-logarithmic factors; that is, this runtime has the same scaling as just collecting the M examples up to the poly-logarithmic factors. The sparse data structure for $|\psi_{\mathrm{in}}\rangle$ is based on the same underlying dense binary tree, but a leaf for each $\{\boldsymbol{x}\}$ stores $\hat{p}_{\mathrm{data}}(\boldsymbol{x}) f(\boldsymbol{x})$ instead of $\hat{p}_{\mathrm{data}}(\boldsymbol{x})$, and each parent store the sum of those at its children in the same way; to construct this data structure, instead of incrementing the counts stored in the nodes by $+1$ for each $\boldsymbol{x}_m$, we add $f(\boldsymbol{x}_m)$ to the number stored in a leaf $\{\boldsymbol{x}_m\}$ and update the numbers stored in the ascendants of this leaf correspondingly.

We use this sparse data structure with QRAM to prepare the states $|\hat{p}_{\mathrm{data}}\rangle$ and $|\psi_{\mathrm{in}}\rangle$. In particular, the preparation of these states is achieved by a parallelized quantum circuit of depth $O(D \log(P))$ to run a quantum algorithm of Grover & Rudolph (2002), where each $O(1)$-depth part of the circuit processes each depth of the $O(D \log(P))$-height tree, and the QRAM is queried once for each of these parts, in total $O(D \log(P))$ times (Kerenidis & Prakash, 2017). Each of the queries to the QRAM is implementable within runtime $O(\operatorname{polylog}(M))$. In particular, the QRAM is an architecture for using classical data in a quantum algorithm without destroying superposition, defined as (1) in the work by Giovannetti et al. (2008b). In our setting, the sparse data structure has at most M nodes as leaves, and the number of nodes at each depth of the tree has at most $1, 2, 4, 8, \ldots, M$ nodes, respectively. Each query to the QRAM performs a quantum circuit depending on one of the collections of these $1, 2, 4, 8, \ldots, M$ nodes at each depth of the tree. The number of nodes to be used in each query is smaller than M. Then, the QRAM for these $O(M)$ nodes is implementable by a $O(\operatorname{polylog}(M))$-depth parallelized quantum circuit on $O(M)$ qubits per query, e.g., by a circuit given in Fig. 10 of the work by Hann et al. (2021), which is queried for each depth of the tree for our sparse data structure. Importantly, the QRAM never measures and reads out classical bit values stored in the $O(M)$ nodes at each depth, but just performs an $O(\operatorname{polylog}(M))$-depth sequence of unitary gates in parallel to maintain quantum superposition. As a whole, given the data structure and QRAM, the overall runtime per preparation of $|\hat{p}_{\mathrm{data}}\rangle$ and $|\psi_{\mathrm{in}}\rangle$ is bounded by

$$O(D \log(P) \times \operatorname{polylog}(M)) = \widetilde{O}(D). \tag{48}$$

To summarize, this runtime is achievable because of the following facts;

- the M examples are stored in the sparse data structure representing the binary tree of height $O(D\log(P))$;
- each depth of the $O(D\log(P))$-height tree is processed by a constant-depth quantum circuit with one query to the QRAM;
- a single query to the QRAM for processing the $O(M)$ nodes at each depth of the tree is implemented by the $O(\mathrm{polylog}(M))$-depth quantum circuit.

With this input model, we provide the proof of Theorem 4.1 on the runtime of Algorithm 2.

Proof. In Algorithm 2, by steps 1, 2, and 3, we prepare a quantum state proportional to

$$\sum_{\boldsymbol{a},b} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)\left|\boldsymbol{a},b\right\rangle = \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{R}\sum_{\boldsymbol{x}}\hat{p}_{\mathrm{data}}(\boldsymbol{x})f(\boldsymbol{x})\left|\boldsymbol{x}\right\rangle, \tag{49}$$

which is the analytical formula for the solution of ridge regression in (18). Then, by step 4 to apply $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I})^{-\frac{1}{2}}$ to this state, we obtain a quantum state

$$\left|p_{\lambda,\Delta}^*\right\rangle = \frac{\sum_{\boldsymbol{a},b} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)\left|\boldsymbol{a},b\right\rangle}{\sqrt{\sum_{\boldsymbol{a},b}|P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b)|^2 + \Delta}}. \tag{50}$$

The runtime of Algorithm 2 is dominated by step 4 requiring

$$\widetilde{O}\left(\frac{D}{\lambda}\frac{1}{\Delta/\gamma}\right), \tag{51}$$

as we will show step by step in the following.

Step 1 for preparing the state $\left|\psi_{\mathrm{in}}\right\rangle$ is performed within runtime

$$\widetilde{O}(D), \tag{52}$$

due to (48).

Step 2 for performing QRT is performed by Algorithm 1 within runtime shown in Theorem 3.1, i.e.,

$$\widetilde{O}(D). \tag{53}$$

Step 3 for applying $(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I})^{-1}$ is performed within runtime $\widetilde{O}(D/\lambda)$, as shown in the following. To explain Step 3, we follow a unified framework for describing a general class of quantum algorithms based on quantum singular value transform (QSVT) developed by Gilyén et al. (2019); in particular, we construct a block encoding of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}$ and implement $(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I})^{-1}$ by applying QSVT to this block encoding. Our construction of the block encoding of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}$ is based on linear combination of block-encoded matrices $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$ and $\boldsymbol{I}$ (Gilyén et al., 2019). The block encoding of $\boldsymbol{I}$ is a trivial unitary operator $\boldsymbol{I}$, and thus we here show the block encoding of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$. For $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$, we use a block encoding of a density operator $\rho = \boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$; in particular, if we have a quantum circuit U for preparing a purification of ρ from $\left|0\right\rangle$, then we can construct a block encoding of ρ using U and $U^\dagger$ a constant number of times (Gilyén et al., 2019). We here prepare the purification of ρ as follows. First, we prepare

$$\sum_{\boldsymbol{x}}\sqrt{\hat{p}_{\mathrm{data}}(\boldsymbol{x})}\left|\boldsymbol{x}\right\rangle \tag{54}$$

within runtime

$$\widetilde{O}(D), \tag{55}$$

due to (48). Then, we add the same number of auxiliary qubits initialized in $\left|0\right\rangle$ as those for the state (54), and apply CNOT gates on the auxiliary qubits controlled by this state to obtain

$$\sum_{\boldsymbol{x}}\sqrt{\hat{p}_{\mathrm{data}}(\boldsymbol{x})}\left|\boldsymbol{x}\right\rangle\otimes\left|\boldsymbol{x}\right\rangle, \tag{56}$$

which requires runtime

$$\widetilde{O}(D) \tag{57}$$

since the number of qubits is $\widetilde{O}(D)$. Then, we apply QRT to the first register to obtain

$$\left(\boldsymbol{R}\sum_{\boldsymbol{x}}\sqrt{\hat{p}_{\mathrm{data}}(\boldsymbol{x})}\left|\boldsymbol{x}\right\rangle\right)\otimes\left|\boldsymbol{x}\right\rangle \tag{58}$$

within runtime shown in Theorem 3.1, i.e.,

$$\widetilde{O}(D). \tag{59}$$

By tracing out the auxiliary qubits, the reduced state of the obtained state is $\rho = \boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$, and the runtime of this block encoding is $\widetilde{O}(D)$. As a result, using the block encodings of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$ and $\boldsymbol{I}$ a constant number of times, the procedure for linear combination of the block encoded matrices $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$ and $\boldsymbol{I}$ yields the block encoding of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}$, which has the runtime

$$\widetilde{O}(D) \tag{60}$$

dominated by that of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top$.

To apply $\left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}\right)^{-1}$ in Step 3 to the state prepared by Step 2, we use QSVT of the block encoding of $\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}$. As shown by Gilyén et al. (2019), given any state $\left|\psi\right\rangle$ and a circuit $\boldsymbol{U}$ for a block encoding of $\boldsymbol{A}$, we can prepare a state proportional to $\boldsymbol{A}^{-1}\left|\psi\right\rangle$ by querying $\boldsymbol{U}$ and $\boldsymbol{U}^\dagger$ in total $\widetilde{O}(\kappa)$ times using the technique of variable-time amplitude amplification (Ambainis, 2012; Childs et al., 2017; Chakraborty et al., 2019; Gilyén et al., 2019), where κ is the condition number of $\boldsymbol{A}$, and the runtime includes that for amplitude amplification. In our case, the condition number of $\boldsymbol{A} = \boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}$ is

$$\kappa \leqq \frac{1+\lambda}{\lambda} = O\left(\frac{1}{\lambda}\right), \tag{61}$$

since the largest eigenvalue $\|\boldsymbol{A}\|_\infty$ of this $\boldsymbol{A}$ is upper bounded by

$$\|\boldsymbol{A}\|_\infty \leqq \|\boldsymbol{R}\|_\infty\|\hat{\boldsymbol{P}}_{\mathrm{data}}\|_\infty\|\boldsymbol{R}^\top\|_\infty + \|\lambda\boldsymbol{I}\|_\infty \leqq 1+\lambda, \tag{62}$$

and the smallest eigenvalue is lower bounded by λ due to the term $\lambda\boldsymbol{I}$. The runtime of each circuit $\boldsymbol{U}$ for this block encoding is $\widetilde{O}(D)$ as shown in (60). As a whole, the runtime by the end of Step 3 is dominated by

$$\widetilde{O}(\kappa)\times\widetilde{O}(D) = \widetilde{O}\left(\frac{D}{\lambda}\right). \tag{63}$$

Step 4 for applying $\left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-\frac{1}{2}}$ is performed within runtime $\widetilde{O}\left(\frac{D}{\lambda}\times\frac{1}{\Delta/\gamma}\right)$, as shown in the following. As in the above explanation, using the framework of QSVT, we here construct a block encoding of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$ and implement $\left(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}\right)^{-\frac{1}{2}}$ by applying QSVT to this block encoding. Our construction of the block encoding of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$ is based on linear combination of block-encoded matrices $\boldsymbol{W}_\lambda$ and $\boldsymbol{I}$. The block encoding of $\boldsymbol{I}$ is a trivial unitary operator, and noticing that $\boldsymbol{W}_\lambda$ is a density operator by definition (i.e., $\boldsymbol{W}_\lambda \geqq 0$ and $\mathrm{Tr}[\boldsymbol{W}_\lambda] = 1$), we show the block encoding of $\boldsymbol{W}_\lambda$ using the block encoding of the density operator similar to the above. In particular, by the end of Step 3, we have constructed a circuit for preparing

$$\frac{1}{\sqrt{\gamma}}\sum_{\boldsymbol{a},b}P^{-\frac{D}{2}}w^*(\boldsymbol{a},b)\left|\boldsymbol{a},b\right\rangle \propto \left(\boldsymbol{R}\hat{\boldsymbol{P}}_{\mathrm{data}}\boldsymbol{R}^\top + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{R}\sum_{\boldsymbol{x}}\sqrt{\hat{p}_{\mathrm{data}}(\boldsymbol{x})}\left|\boldsymbol{x}\right\rangle, \tag{64}$$

which runs within time $\widetilde{O}(\frac{D}{\lambda})$ due to (63). Then, in the same way as (56), we add the same number of auxiliary qubits initialized in $\left|0\right\rangle$ as those for this state, and apply CNOT gates on the auxiliary qubits controlled by the above state to obtain

$$\frac{1}{\sqrt{\gamma}}\sum_{\boldsymbol{a},b}P^{-\frac{D}{2}}w^*(\boldsymbol{a},b)\left|\boldsymbol{a},b\right\rangle\otimes\left|\boldsymbol{a},b\right\rangle, \tag{65}$$

Algorithm 3 Quantum algorithm for finding a winning ticket for the original neural network $\mathcal{S}[w_\lambda^*]$.

```
Require: $\epsilon, \delta, \lambda > 0$, assumptions in Sec. 4.1, $M$ examples stored in data structure in Sec 4.2, $\Delta$ and $N$ bounded by Theorem 4.2.
Ensure: A subnetwork $\hat{f}$ of $\mathcal{S}[w_\lambda^*]$ achieving (21) and characterized as (26) by the set of parameters $\hat{\mathbb{W}} \subset \mathbb{Z}_P^D \times \mathbb{Z}_P$ and the weights $\hat{\boldsymbol{w}}^*$ for the nodes in the hidden layer.
1: Initialize $\hat{\mathbb{W}} = \emptyset$.
2: for $N$ times do
3:     Sample $(\boldsymbol{a}, b) \in \mathbb{Z}_P^D \times \mathbb{Z}_P$ from $p_{\lambda,\Delta}^*(\boldsymbol{a}, b)$ in (23) by the quantum algorithm in Theorem 4.1.
4:     Add the sampled parameter $(\boldsymbol{a}, b)$ to $\hat{\mathbb{W}}$.
5: end for
6: Train the weight $w(\boldsymbol{a}, b)$ of the sampled subnetwork $\sum_{(\boldsymbol{a},b)\in\hat{\mathbb{W}}} w(\boldsymbol{a}, b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P)$ by convex optimization using the given $M$ examples, to obtain $\hat{\boldsymbol{w}}^*$ as a solution of minimizing $\tilde{J}(w)$ in (18). {This convex optimization can be performed by classical algorithms such as stochastic gradient descent (SGD) (Harvey et al., 2019).}
7: Return $\hat{\mathbb{W}}$ and $\hat{\boldsymbol{w}}^*$.
```

which requires $\widetilde{O}(D)$ runtime since the number of qubits is $\widetilde{O}(D)$. By tracing out the auxiliary qubits, the reduced state becomes

$$\boldsymbol{W}_\lambda = \sum_{\boldsymbol{a},b} \frac{|P^{-\frac{D}{2}} w^*(\boldsymbol{a}, b)|^2}{\gamma} |\boldsymbol{a}, b\rangle \langle \boldsymbol{a}, b| \,, \tag{66}$$

and the runtime of this block encoding is $\widetilde{O}(D/\lambda)$ dominated by the state-preparation circuit by the end of Step 3. Then, using the block encodings of $\boldsymbol{W}_\lambda$ and $\boldsymbol{I}$ a constant number of times, the linear combination of the block encoded matrices $\boldsymbol{W}_\lambda$ and $\boldsymbol{I}$ yields the block encoding of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$, which has the runtime

$$\widetilde{O}\Big(\frac{D}{\lambda}\Big). \tag{67}$$

To apply $(\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma})\boldsymbol{I}^{-\frac{1}{2}}$ in Step 4, we use QSVT of the block encoding of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$. In the same way as applying $\boldsymbol{A}^{-1}$ explained above, given any state $|\psi\rangle$ and a circuit $\boldsymbol{U}$ for a block encoding of $\boldsymbol{A}$, using the technique of variable-time amplitude amplification, we can prepare a state proportional to $\boldsymbol{A}^{-\frac{1}{2}} |\psi\rangle$ by querying $\boldsymbol{U}$ and $\boldsymbol{U}^\dagger$ in total $\widetilde{O}(\kappa)$ times (Gilyén et al., 2019). In the same way as the analysis of Step 3 with replacing λ with $\frac{\Delta}{\gamma}$, the condition number of $\boldsymbol{W}_\lambda + \frac{\Delta}{\gamma}\boldsymbol{I}$ is $O(\frac{1}{\Delta/\gamma})$. As a whole, the runtime by the end of Step 4 is dominated by

$$\widetilde{O}\Big(\frac{1}{\Delta/\gamma}\Big) \times \widetilde{O}\Big(\frac{D}{\lambda}\Big). \tag{68}$$

Consequently, the overall runtime of Algorithm 2 is bounded by (68) in Step 4, i.e.,

$$\widetilde{O}\left(\frac{D}{\lambda}\frac{1}{\Delta/\gamma}\right) = \widetilde{O}\left(\frac{D}{\lambda\Delta} \times \gamma\right), \tag{69}$$

which yields the conclusion. □

E. Proof on Bounds for Finding Winning Ticket of Neural Networks

We provide the proof of Theorem 4.2 in Sec. 4.3. Our algorithm for Theorem 4.2 is shown in Algorithm 3.

Proof of Theorem 4.2. With writing

$$N_\epsilon \coloneqq \Big(\frac{\alpha}{\beta\sqrt{\epsilon}}\Big)^{\frac{1}{\beta}}, \tag{70}$$

we set Δ and N as

$$\Delta = \left(\alpha(N_\epsilon + 1)^{-(1+\beta)}\right)^2 = \Omega(\epsilon^{1+\frac{1}{\beta}}), \tag{71}$$

$$N=\left\lceil 2\left(\lceil N_\epsilon\rceil+\frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}\right)\ln\left(\frac{\lceil N_\epsilon\rceil}{\delta}\right)\right\rceil=O\Big(N_\epsilon\log\Big(\frac{N_\epsilon}{\delta}\Big)\Big)=O\left(\frac{1}{\epsilon^{\frac{1}{2\beta}}}\log\Big(\frac{1}{\epsilon\delta}\Big)\right). \tag{72}$$

Note that we can set Δ and N flexibly up to changing α and β in the constant factors of (71) and (72), as long as the function f to be learned remains in the set of (α,β)-class functions. The parameter N_ϵ is chosen in such a way that we have

$$\sum_{j=\lceil N_\epsilon\rceil+1}^{\infty}\alpha j^{-(1+\beta)}\leqq\int_{N_\epsilon}^{\infty}\alpha x^{-(1+\beta)}dx=\frac{\alpha}{\beta N_\epsilon^{\beta}}=\sqrt{\epsilon}. \tag{73}$$

For the analysis, let $\mathbb{W}_{\lambda,\Delta}$ denote a set of parameters $(\boldsymbol{a},b)$ for high-weight nodes in the hidden layer of $\mathcal{S}[w_\lambda^*]$, given by

$$\mathbb{W}_{\lambda,\Delta}\coloneqq\{(\boldsymbol{a}_j,b_j)\in\mathbb{Z}_P^D\times\mathbb{Z}_P:|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a}_j,b_j)|^2\geqq\Delta,j\in\{1,\ldots,P^{D+1}\}\}. \tag{74}$$

Then, due to the assumption on the (α,β)-class functions

$$|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a}_j,b_j)|\leqq\alpha j^{-(1+\beta)}, \tag{75}$$

we have by construction

$$|\mathbb{W}_{\lambda,\Delta}|\leqq\lceil N_\epsilon\rceil. \tag{76}$$

For any $(\boldsymbol{a},b)\in\mathbb{W}_{\lambda,\Delta}$, we have

$$\Pr\{(\boldsymbol{a},b)\notin\hat{\mathbb{W}}\} \tag{77}$$
$$=(1-p_{\lambda,\Delta}^*(\boldsymbol{a},b))^N \tag{78}$$
$$=\Big(1-\frac{1}{Z}\frac{|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a},b)|^2}{|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a},b)|^2+\Delta}\Big)^N \tag{79}$$
$$\leqq\Big(1-\frac{1}{|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}}\frac{\Delta}{\Delta+\Delta}\Big)^N \tag{80}$$
$$\leqq\exp\left(-\frac{N}{2\Big(|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}\Big)}\right), \tag{81}$$

where the first inequality follows from

$$Z=\sum_{(\boldsymbol{a},b)\in\mathbb{W}_{\lambda,\Delta}}\frac{|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a},b)|^2}{|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a},b)|^2+\Delta}+\sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P\setminus\mathbb{W}_{\lambda,\Delta}}\frac{|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a},b)|^2}{|P^{-\frac{D}{2}}w_\lambda^*(\boldsymbol{a},b)|^2+\Delta} \tag{82}$$
$$\leqq|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{\Delta}\sum_{j=\lceil N_\epsilon\rceil+1}^{\infty}\left|\alpha j^{-(1+\beta)}\right|^2 \tag{83}$$
$$=|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{\Delta}\sum_{j=\lceil N_\epsilon\rceil+1}^{\infty}\alpha^2 j^{-2(1+\beta)} \tag{84}$$
$$\leqq|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{\Delta}\int_{N_\epsilon}^{\infty}\frac{\alpha^2}{x^{-2(1+\beta)}}dx \tag{85}$$
$$=|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{\Delta}\frac{\alpha^2}{(1+2\beta)N_\epsilon^{1+2\beta}} \tag{86}$$
$$=|\mathbb{W}_{\lambda,\Delta}|+\frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}, \tag{87}$$

and the second inequality follows from $1-x\leqq\mathrm{e}^{-x}$ for $x\in\mathbb{R}$. Thus, due to the union bound, it follows from (72) and (76) that

$$\Pr\{\mathbb{W}_{\lambda,\Delta}\nsubseteqq\hat{\mathbb{W}}\} \tag{88}$$

$$\leqq |\mathbb{W}_{\lambda,\Delta}| \exp\left(-\frac{N}{2\left(|\mathbb{W}_{\lambda,\Delta}| + \frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}\right)}\right) \tag{89}$$

$$\leqq \delta \frac{|\mathbb{W}_{\lambda,\Delta}|}{\lceil N_\epsilon \rceil} \exp\left(-\frac{2\left(\lceil N_\epsilon \rceil + \frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}\right)}{2\left(|\mathbb{W}_{\lambda,\Delta}| + \frac{1}{1+2\beta}\frac{(N_\epsilon+1)^{2+2\beta}}{N_\epsilon^{1+2\beta}}\right)}\right) \tag{90}$$

$$\leqq \delta. \tag{91}$$

Therefore, with the choice of N in (72), we have $\mathbb{W}_{\lambda,\Delta} \subseteqq \hat{\mathbb{W}}$ with high probability greater than $1-\delta$. Then for any $\hat{\mathbb{W}}$ satisfying $\mathbb{W}_{\lambda,\Delta} \subseteqq \hat{\mathbb{W}}$, the subnetwork with nodes in the hidden layer parameterized by $(\boldsymbol{a}, b) \in \hat{\mathbb{W}}$ approximates the original network $\mathcal{S}[w_\lambda^*]$ as

$$\inf_w \left\{ \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) \left| \mathcal{S}[w_\lambda^*](\boldsymbol{x}) - \sum_{(\boldsymbol{a},b)\in\hat{\mathbb{W}}} P^{-\frac{D}{2}} w(\boldsymbol{a},b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) \right|^2 \right\} \tag{92}$$

$$\leqq \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) \left| \mathcal{S}[w_\lambda^*](\boldsymbol{x}) - \sum_{(\boldsymbol{a},b)\in\mathbb{W}_{\lambda,\Delta}} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) \right|^2 \tag{93}$$

$$= \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) \left| \sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) - \sum_{(\boldsymbol{a},b)\in\mathbb{W}_{\lambda,\Delta}} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) \right|^2 \tag{94}$$

$$= \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) \left| \sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P\setminus\mathbb{W}_{\lambda,\Delta}} P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) \right|^2 \tag{95}$$

$$\leqq \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) \left(\sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P\setminus\mathbb{W}_{\lambda,\Delta}} \left| P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) \right| \left| g((\boldsymbol{a}^\top \boldsymbol{x} - b) \bmod P) \right| \right)^2 \tag{96}$$

$$\leqq \sum_{\boldsymbol{x}} \hat{p}_{\mathrm{data}}(\boldsymbol{x}) \left(\sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P\setminus\mathbb{W}_{\lambda,\Delta}} \left| P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) \right| \right)^2 \tag{97}$$

$$= \left(\sum_{(\boldsymbol{a},b)\in\mathbb{Z}_P^D\times\mathbb{Z}_P\setminus\mathbb{W}_{\lambda,\Delta}} \left| P^{-\frac{D}{2}} w_\lambda^*(\boldsymbol{a},b) \right| \right)^2 \tag{98}$$

$$\leqq \left(\sum_{j=\lceil N_\epsilon \rceil+1}^{\infty} \alpha j^{-(1+\beta)} \right)^2 \tag{99}$$

$$\leqq \epsilon, \tag{100}$$

which yield the conclusion. □

F. Detail of Numerical Experiment on Advantage of Using Quantum Ridgelet Transform

We describe the detail of the parameters chosen for the numerical experiment in Sec. 4.4.

In our numerical experiment, we fixed

$$P = 127, \tag{101}$$

$$D = 1. \tag{102}$$

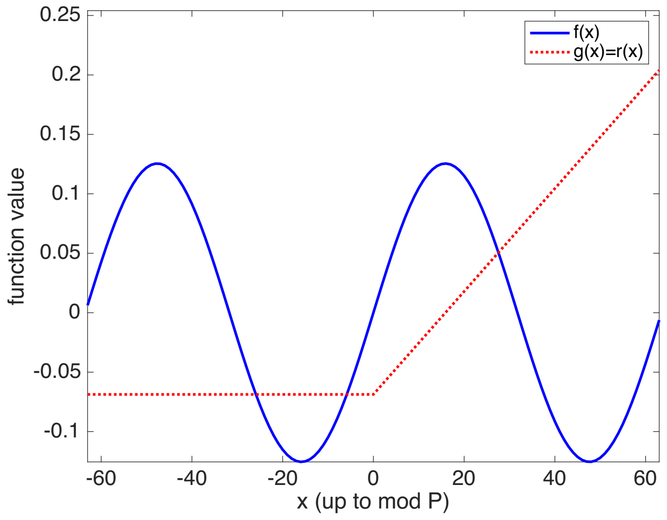

Figure 2. The function f to be learned and the activation function g chosen as ReLU, where the ridgelet function r is also chosen as $r = g$ in our numerical experiment.

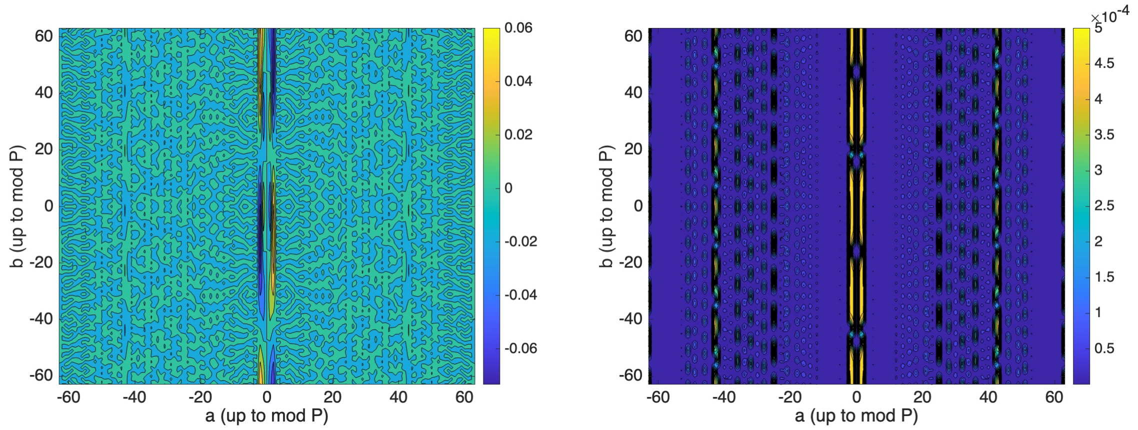

Figure 3. The discrete ridgelet transform $\mathcal{R}[f](a, b)$ of f in Fig. 2 on the left, and the optimized probability distribution $p^*_{\lambda,\Delta}(a, b)$ on the right.

In the current case of $D = 1$, we may write $\boldsymbol{x}$ and $\boldsymbol{a}$ as x and a, respectively. Up to normalization, we chose

$$f(x) \propto \sin\left(\frac{4\pi x}{P}\right), \tag{103}$$

$$g(x) = r(x) \propto \begin{cases} x \bmod P & 0 \leqq (x \bmod P) \leqq \frac{P-1}{2}, \\ 0 & \frac{P+1}{2} \leqq (x \bmod P) \leqq P-1, \end{cases} \tag{104}$$

which are shown in Fig. 2 with normalization. The empirical distribution was chosen as the uniform distribution

$$\hat{p}_{\mathrm{data}}(x) = \frac{1}{P}. \tag{105}$$

The parameters in our setting were chosen as

$$\epsilon = 5 \times 10^{-2}, \tag{106}$$

$$\lambda = 1 \times 10^{-4}, \tag{107}$$

$$\alpha = 4 \times 10^{21}, \tag{108}$$

$$\beta = 5, \tag{109}$$

which leads to

$$\gamma \approx 9.8 \times 10^{-1}. \tag{110}$$

According to (70), (71), and (72), it suffices to set

$$N_\epsilon \approx 2.0 \times 10^{4}, \tag{111}$$

$$\Delta \approx 5.5 \times 10^{-5}, \tag{112}$$

$$N \approx 5.9 \times 10^{5}. \tag{113}$$

With the above choice of parameters, the discrete ridgelet transform $\mathcal{R}[f](a,b)$ of f and the optimized probability distribution $p^*_{\lambda,\Delta}(a,b)$ are calculated and illustrated in Fig. 3. In our numerical experiment, the sampling from $p^*_{\lambda,\Delta}$ is classically simulated by rejection sampling using the calculated value of $p^*_{\lambda,\Delta}(a,b)$ for each a and b. The uniform distribution $p_{\mathrm{uniform}}(a,b) = \frac{1}{P^{D+1}}$ is used for sampling the random features. After performing the sampling N times for $N = 4, 8, \ldots, 120$, we optimize $\hat{w}^*(a,b)$ in (26) by minimizing the empirical risk $\sum_x \hat{p}_{\mathrm{data}}(x)|f(x) - \sum_{(a,b)\in\hat{\mathbb{W}}} P^{-\frac{D}{2}} \hat{w}^*(a,b) g((ax-b) \bmod P)|^2$. This convex optimization was solved by MOSEK (ApS, 2022) and YALMIP (Löfberg, 2004). The result of this numerical experiment is presented in Fig. 1 of Sec. 4.4 in the main text.

To support our results further, we also performed numerical simulation in the case of choosing $g\ (= r)$ as a sigmoid function rather than ReLU used in Fig. 1; in particular, in this additional numerical simulation, with the same f as (103), we chose, up to normalization,

$$g(x) = r(x) \propto \tanh\left(\frac{10x}{P}\right), \tag{114}$$

which are shown at the top of Fig. 4 with normalization. Setting the other parameters the same as Fig. 1, we also present the result of this additional numerical experiment at the bottom of Fig. 4.

These numerical results show that the actual performance of empirical risk minimization in our numerical experiments can be much better than the theoretically guaranteed upper bound on N in Theorem 4.2, implying even more actual advantage of our algorithm than the theoretically guaranteed bound in our setting.

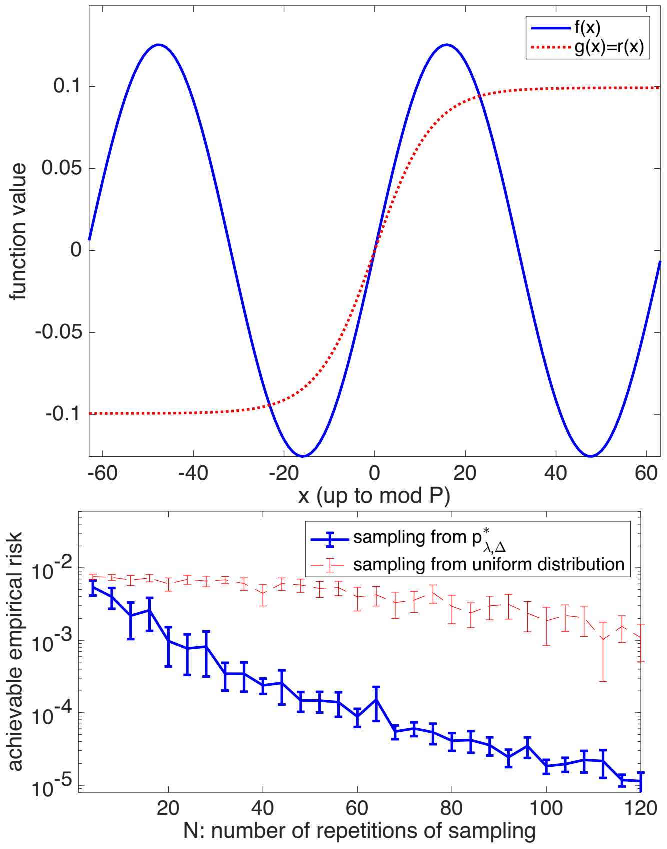

Figure 4. The numerical experiment performed in addition to the one presented in Fig. 1 of the main text. In this additional numerical experiment, as shown at the top of the figure, the function f to be learned is the same as the one in Fig. 2, whereas the activation function g and the ridgelet function r are chosen as a sigmoid function (tanh) as shown in the figure, rather than ReLU in Fig. 2. Setting the other parameters the same as the numerical experiment in Fig. 1, at the bottom of the figure, we plot the empirical risks achieved by sampling from the optimized distribution via our algorithm (blue thick line) and that from the uniform distribution via random features (red dashed line), which support our results further in the same way as Fig. 1.