

Bernstein–Vazirani Networks: Quantum Machine Learning by Interference

Natacha Kuete Meli¹ Tolga Birdal² Prayag Tiwari³ Vladislav Golyanik^{4,†} Michael Moeller^{1,†}

¹*Department of Vision and Graphics,
University of Siegen*

³*School of Information Technology,
Halmstad University*

²*Department of Computing,
Imperial College London*

⁴*4D and Quantum Vision,
Max Planck Institute for Informatics*

Abstract

We introduce Bernstein–Vazirani Networks (BVNs), a non-variational quantum machine learning framework that leverages quantum interference for supervised learning, demonstrated on vision and representation learning tasks. In their standard form, BVNs follow the principle of quantum Fourier sampling: labelled data are placed in superposition and interfered in the Fourier basis to extract globally informative features. We then define generalised BVNs that enable interference in problem-adapted bases, yielding more expressive models under the same measurement budget as in the standard setting. BVNs achieve universal function approximation through (over)complete interference bases, while training of BVNs is gradient-free. Experiments on synthetic and real-world classification tasks, as well as implicit image representation, show strong generalisation capabilities and competitive performance with classical and quantum baselines¹.

1. Introduction

Quantum Machine Learning (QML) (Biamonte et al., 2017; Cerezo et al., 2022) aims to exploit the high-dimensional Hilbert spaces of quantum systems coupled with quantum effects to build expressively powerful learning models. Most existing QML approaches typically rely on Parametrised Quantum Circuits (PQCs) and face significant architectural and trainability challenges. A central challenge is the mismatch between classical neural networks’ reliance on non-linear activations and the linear, unitary dynamics of closed quantum systems, where nonlinearity is recoverable only through Boolean encodings, as noted by Herman et al. (Herman et al., 2023). In addition, PQC optimisation is often

hindered by barren plateaus, i.e., regions of the loss landscape where gradients vanish exponentially with system size (McClean et al., 2018), and, even outside such regimes, by the complexity of certified gradient-evaluation methods (e.g., parameter-shift rules (Schuld et al., 2019)) combined with measurement noise, imposing substantial overhead in the execution and sampling of quantum circuits. A recent study (Recio-Armengol et al., 2025) further confirms this crisis in PQCs, *estimating to 38 years the runtime of a single epoch of gradient descent with the parameter-shift rule on MNIST with a 10^4 -parameter PQC, requiring $\sim 10^{15}$ measurement shots*. These recent insights expose a tangible gap between the theoretical expressivity of quantum models and their practical efficiency to emulate neural networks, motivating alternative QML paradigms.

From this perspective, it is instructive to revisit quantum algorithms that originally demonstrated provable advantages compared to classical counterparts, such as Shor (Shor, 1999), Deutsch–Jozsa (Deutsch & Jozsa, 1992) or Bernstein–Vazirani (Bernstein & Vazirani, 1993), which use quantum Fourier sampling to extract hidden structure via *interference*. Wakeham and Schuld (Wakeham & Schuld, 2024) identify a common blueprint underlying such algorithms that can be repurposed for inference tasks: *place labelled information in superposition via an oracle, interfere in the Fourier space, and measure*. In this paradigm, learning is achieved via interference by sampling globally informative Fourier characters rather than via gradient-based optimisation, avoiding issues such as local minima and barren plateaus. This reframes learning alternatively as a problem of frequency discovery rather than parameter optimisation. Wakeham and Schuld, however, left unresolved the “leakage” problem that occurs when no single frequency fits the data perfectly, preventing them from validating the idea practically.

Building on this blueprint, we introduce *Bernstein–Vazirani Networks (BVNs)*, which *repurpose* the Bernstein–Vazirani (BV) algorithm (Bernstein & Vazirani, 1993) for learning tasks. The BV algorithm interferes labelled training exam-

¹Project page: <https://4dqv.mpi-inf.mpg.de/BVNs/>

[†]denotes equal advising

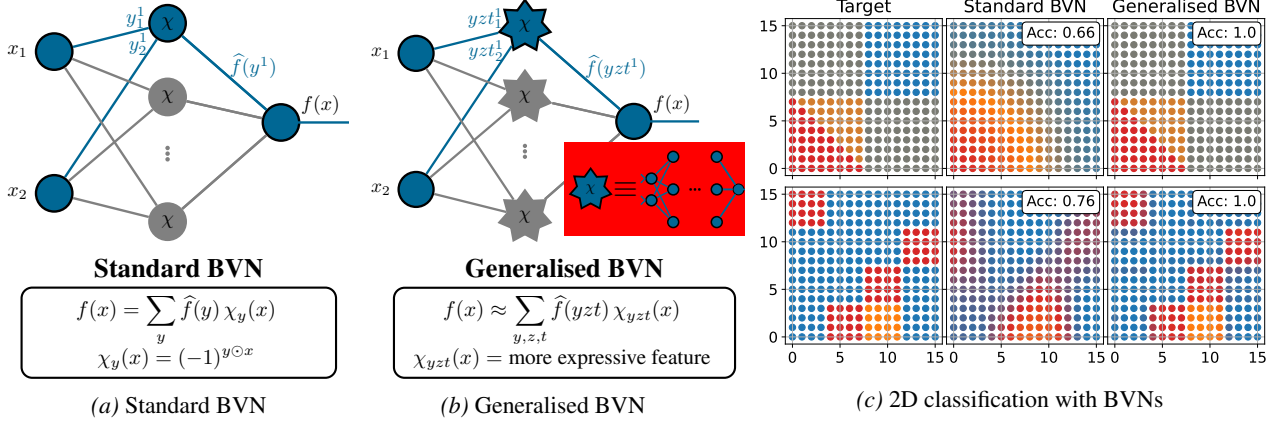


Figure 1. The proposed standard (1a) and generalised (1b) Bernstein–Vazirani Networks (BVNs). In the standard setting, binary inputs x_i are linearly combined in a hidden layer with optimised binary weights y^j , passed through a parity activation χ , and linearly combined at the output layer using Fourier coefficients $\hat{f}(y^j)$. In the generalised setting, inputs pass through expressive optimised sub-models χ (cartooned in the red box) before the output combination, enabling richer representations. (1c): Illustratively, we classify 2D points where the target function (left) is a sum of expressive sub-models χ of the generalised basis. The generalised model fits the boundaries exactly, while the standard model has not yet converged. “Acc.” denotes accuracy. See Secs. 4 and 5 for more details.

ples placed in superposition to reveal a set of basis functions that explain the training data. We further generalise the standard BV framework—defining *generalised BVNs*—by allowing interference in alternative problem-adapted bases, enabling to sample more diverse and expressive features while reducing the required measurement budget. Importantly, quantum circuits in BVNs require no variational parameter optimisation with respect to a loss function and rely solely on labelled data to infer the underlying target function that explains the observations. The expressivity of our models stems from the exponentially large Hilbert space and the (over)completeness of the interference basis that enable universal function approximation.

Intuitively, generalised BVNs can be viewed as expressive parametrised quantum architectures, *in which all network parameter configurations are placed in superposition*. Interference then highlights the configurations that best align with the structure induced by the labelled data. Rather than identifying a single optimal configuration, the algorithm samples multiple high-quality configurations, which are aggregated to approximate the target function, addressing the leakage problem. Each sampled configuration corresponds to a sub-network within a larger implicit ensemble, evaluated at the cost of a single sub-network instance and optimised through interference rather than gradient descent. An intuitive illustration of the proposed BVNs is shown in Figure 1 with a hand-crafted 2D point classification result, where the target function is a sum of basis functions unique to the generalised basis. After only 100 shots, the generalised BVN fits the decision boundaries precisely, whereas the not-yet-converged standard BVN smooths them.

In summary, our key technical contributions are as follows:

- **Bernstein–Vazirani Networks (BVNs)**, a new class of QML models that leverage quantum interference to discover dominant basis functions for efficient function approximation in machine learning tasks;
- **Generalised BVNs** that extend the standard framework with more expressive basis functions, improving data fitting under the same measurement budget.
- **Coefficient reconstruction via ridge regression** to address the leakage problem and recover the best approximation of the target function in the sampled basis.

Empirically, we validate our BVNs on several synthetic and real-world classification tasks (Iris (Fisher, 1936) and Penguins (Horst et al., 2022) datasets), as well as 2D image-fitting tasks. Classification results demonstrate the ability of BVNs to learn decision boundaries and perform on par with classical (MLP, SVM) and quantum models (Single-Qubit Classifier (Pérez-Salinas et al., 2020)), while being far more sampling-efficient than PQC models. In line with this, our BVNs continue to perform accurately on image representation tasks, competing with classical models (MLP+RFF, SIREN (Benbarka et al., 2022)) and outperforming the recent PQC models (QIREN (Zhao et al., 2024) and QVF (Wang et al., 2025)) in both accuracy and sampling cost.

2. Related Works

Quantum Machine Learning (QML) (Biamonte et al., 2017; Schuld & Petruccione, 2018; Cerezo et al., 2022) is driven by the idea of combining the power of quantum computing with the success of neural networks. We review key developments in the field and refer the reader to a recent survey for further details (Meli et al., 2025).

Quantum Neural Network Designs. Schuld et al. (Schuld et al., 2014), by surveying early work on Quantum Neural Networks (QNNs), outlined three requirements for QNNs: (i) a fixed-length bitstring initial state; (ii) neural-like connections and update rules; and (iii) quantum-consistent evolution exploiting effects such as superposition, entanglement, and interference. These requirements motivated several elementary QNN architectures, including quantum neurons with ancillary registers (Cao et al., 2017) and various quantum perceptron models (Beer et al., 2020; da Silva et al., 2016; Kapoor et al., 2016). While replacing classical components with quantum counterparts is a reasonable direction, a central challenge in QML has long been the tension between the nonlinear, dissipative dynamics of neural computation and the linear, unitary dynamics of quantum mechanics (Schuld et al., 2014). Herman et al. (Herman et al., 2023) showed that one way to bypass this issue is to map inputs to the Boolean cube. Yet, training their models still relies on a classical optimisation loop.

Recently, Wakeham and Schuld (Wakeham & Schuld, 2024) explored an alternative approach to QML drawn from quantum algorithms such as Deutsch–Jozsa (Deutsch & Jozsa, 1992) and Shor’s algorithm (Shor, 1999), rather than from classical neural networks. They highlight a common blueprint shared by these algorithms: *place labelled information in superposition via an oracle, interfere in Fourier space, and measure*. The authors then leverage this *interference strategy for inference*, by regarding a learning task as a hidden-subgroup problem in which interference should reveal the so-called “annihilator” that generated the data, i.e., the true oracle that produced the training examples. This perspective suggests a promising route toward genuinely quantum learning models by directly aligning inference with intrinsic quantum mechanisms. Wakeham and Schuld (Wakeham & Schuld, 2024) left open how to handle leakage, i.e., loss of information when no perfect annihilator exists. We address this gap by drawing on the Bernstein–Vazirani algorithm (Bernstein & Vazirani, 1993), validating the approach on several learning tasks and showing how “leaked information” can be aggregated.

Variational Quantum Neural Networks. Variational quantum neural networks implemented via Parametrised Quantum Circuits (PQCs) are the dominant paradigm in contemporary QML. Schuld and Petruccione (Schuld & Petruccione, 2018) formalised supervised learning with quantum computers, introducing the embedding of classical data into quantum states and optimising PQCs as trainable models. Key developments include PQC architectures designed for specific expressivities and inductive biases. Notably, data re-uploading classifiers showed that a single qubit with repeated data encoding suffices for binary classification

(Pérez-Salinas et al., 2020), later formalised as implementing a Fourier series (Schuld et al., 2021). Further work analysed how entanglement patterns affect expressivity (Sim et al., 2019) and introduced quantum convolutional neural networks inspired by classical ones (Cong et al., 2019).

A major challenge for PQC is barren plateaus (Larocca et al., 2025), where flat loss landscapes hinder optimisation. Only a few architectures, mostly group-equivariant models with built-in inductive biases (Schatzki et al., 2024; Liu et al., 2025; Nguyen et al., 2024) or shallow-depth PQCs, admit theoretical training guarantees. Our approach diverges fundamentally from this paradigm by eliminating classical outer-loop optimisation and instead exploiting quantum interference to determine network configurations.

Applications. Representative supervised learning tasks for assessing QML models include classification and implicit representation. For classification, Salinas et al. introduced the Single-Qubit Classifier (SQC) (Pérez-Salinas et al., 2020), which associates target classes with specific regions of the Bloch sphere and trains PQCs to cluster input states in the corresponding class representations. The SQC employs data re-uploading to approximate functions through their Fourier representations. This Fourier-based perspective was later extended to Quantum Implicit Neural Representation (QIREN) by Zhao et al. (Zhao et al., 2024), where PQCs learn a mapping from coordinates to signal values, such as those describing images or audio. More recently, Wang et al. (Wang et al., 2025) introduced Quantum Visual Fields (QVFs), a hybrid architecture in which a classical backbone extracts and normalises feature vectors that are subsequently processed by PQCs, improving the expressive capacity of the quantum model.

3. Motivation

Supervised learning (Nielsen, 2015; Goodfellow et al., 2016) seeks to approximate an unknown function f from training pairs $\{(x^i, y^i)\}_{i=1}^m$, where $f(x^i) = y^i$. This is typically achieved by a parameterised model $\mathcal{N}$ such that $\mathcal{N}(x^i, \theta) \approx f(x^i)$ for some optimal parameter set θ . This network $\mathcal{N}$ is often organised into layers, with an output layer aggregating the results of previous layers. We can express this as

$$\mathcal{N}(x) = \sum_{j=1}^k w_j \phi_j(x), \quad (1)$$

where $\phi_j(x)$ denotes the output of the j -th neuron in the last hidden layer and w_j are the corresponding output weights (both collectively parametrised by θ). In essence, we are representing the target function f in a learned basis formed by the functions $\{\phi_j\}_{j=1}^k$ produced before the output layer.

Typically, training requires designing an expressive archi-

Table 1. List of symbols used in this paper.

$\mathbb{Z}_2^n$	n -dimensional Boolean space, $\mathbb{Z}_2^n = \{0, 1\}^n$
$\odot, \oplus$	inner-product and sum mod2, respectively
$\tilde{f}, f$	standard and generalised target function
$\hat{f}(\cdot)$	Fourier coefficient of f
H	Hadamard transform
I	interference operator
U_f	oracle unitary for function f
η	expressive representation
ξ, χ	standard and generalised basis function
$\langle , \rangle$	bra and ket, complex row and column vectors

tecture $\mathcal{N}$, performing multiple passes over the data, and optimising non-trivial loss functions. The expressive architecture is constructed so that the hidden layers, or basis functions $\{\phi_j\}_{j=1}^k$, lift the data into a higher-dimensional space, which in practice increases the number of parameters. For this reason, it is believed that QC, which can access high-dimensional Hilbert spaces, may offer advantages in ML, giving rise to QML (Biamonte et al., 2017; Schuld et al., 2014; Cerezo et al., 2022).

An interesting question we would like to answer is how to design a learning paradigm that, given access to the training dataset, can infer within a high-dimensional basis $\{\phi_j\}$ a reasonably sized subset $\{\phi_j\}_{j \in S}$, $|S| \ll |\{\phi_j\}|$, that approximates f via Equation (1) sufficiently well.

Our BVNs operate at the natural intersection of learning and quantum computing in the sense that, by exploiting superposition and interference, they sample indexed basis functions from an exponentially large basis with probability proportional to the amplitudes of their weights in the representation of the target function f in that basis. These properties, together with the fact that BVNs (Section 4) avoid several fundamental limitations of gradient-based PQC training, make them a highly appealing learning paradigm.

4. (Generalised) Bernstein–Vazirani Networks

We introduce our interference-based models, beginning with a warm-up on the Bernstein–Vazirani algorithm. Key symbols and notations used in this work are listed in Table 1.

4.1. Warm-up: The Bernstein–Vazirani Algorithm

The Bernstein–Vazirani (BV) algorithm (Bernstein & Vazirani, 1993) is a celebrated example of quantum advantage. It solves the following problem: Given a Boolean function $\tilde{f} : \mathbb{Z}_2^n \rightarrow \{0, 1\}$ that is *promised* to satisfy $\tilde{f}(x) = s \odot x = s^\top x \bmod 2$ for an *unknown* $s \in \mathbb{Z}_2^n$, the goal is to recover s from observed pairs $(x, \tilde{f}(x))$.

The algorithm, illustrated in Figure 2, has three steps: (i) starting from $|0\rangle^{\otimes n}$, we apply $H^{\otimes n}$ to create a uniform

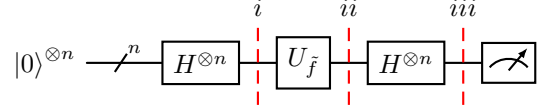


Figure 2. The Bernstein–Vazirani algorithm (Bernstein & Vazirani, 1993). The algorithm mainly consists of three steps: (i) initialise the system in a perfect superposition; (ii) call the function oracle; (iii) apply the Hadamard operator for interference. If $\tilde{f}$ satisfies the promise $\tilde{f}(x) = s \odot x$, measuring reveals s with certainty.

superposition over all basis states, i.e., inputs; (ii) we query the oracle $U_{\tilde{f}}$, which applies the phase $(-1)^{\tilde{f}(x)}$ to each basis state in the superposition; (iii) we apply $H^{\otimes n}$ again to interfere with the phases. The corresponding states are

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} |x\rangle, \quad (2)$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} (-1)^{\tilde{f}(x)} |x\rangle, \quad (3)$$

$$|\psi_3\rangle = \sum_{y \in \mathbb{Z}_2^n} \hat{f}(y) |y\rangle, \quad \hat{f}(y) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{\tilde{f}(x) + y \odot x}. \quad (4)$$

Substituting the promise $\tilde{f}(x) = s \odot x$ into Equation (4) gives $\hat{f}(y) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(s \oplus y) \odot x}$, which equals 1 if and only if $y = s$, hence 0 else (see Appendix A). In essence, thanks to interference, a single measurement of the system reveals s out of 2^n solutions with probability 1.

Quantum Fourier Sampling. The BV algorithm (Bernstein & Vazirani, 1993) generalises to Fourier sampling over $\mathbb{Z}_2^n$: If the promise $\tilde{f}(x) = s \odot x$ holds, we measure s with probability 1. If not, consider the function space $\mathcal{F} := \{f : \mathbb{Z}_2^n \rightarrow \mathbb{R}\}$ and the functions $\chi_y(x) = (-1)^{y \odot x}$, parameterised by $y \in \mathbb{Z}_2^n$, which encode the parity $y \odot x$. The set $\{\chi_y\}_{y \in \mathbb{Z}_2^n}$ forms an orthonormal basis of $\mathcal{F}$ with respect to the inner product $\langle f, g \rangle = \mathbb{E}_{x \in \mathbb{Z}_2^n} [f(x)g(x)]$, and we refer to $y \in \mathbb{Z}_2^n$ as a basis vector. It follows that

$$\hat{f}(y) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} f(x) \chi_y(x) \quad (5)$$

are precisely the Fourier coefficients of f in this basis χ .

Theorem 4.1 (Fourier expansion on $\mathbb{Z}_2^n$, (O’Donnell, 2014)). *Any function $f : \mathbb{Z}_2^n \rightarrow \mathbb{R}$ admits a unique expansion as*

$$f(x) = \sum_{y \in \mathbb{Z}_2^n} \hat{f}(y) \chi_y(x). \quad (6)$$

This is called the Fourier expansion of f , and the coefficients $\hat{f}(y) \in \mathbb{R}$ are the Fourier coefficients of f in the basis χ .

Proof. See Appendix B. $\square$

As implied by Theorem 4.1 and Equations (2) to (4), running the BV algorithm for $f(x) = (-1)^{\tilde{f}(x)}$ or any other f yields

basis vectors $y \in \mathbb{Z}_2^n$ and corresponding functions $\chi_y \in \mathcal{F}$ that approximate f . Moreover, each y is sampled with probability equal to the squared of its Fourier coefficient $|\hat{f}(y)|^2$, so components with large coefficients appear first. This operation is known as Fourier sampling (Bernstein & Vazirani, 1993). An alternative linear-algebraic derivation of BVNs is given in Appendix C.

A Neural View on the Bernstein–Vazirani Model. We can view the BV algorithm as a two-layer network, which we call a *Bernstein–Vazirani Network (BVN)* and depict in Figure 1a. Hidden units compute parity activations $\chi_y(x) = (-1)^{y \odot x}$ on binary inputs x_i weighted by y_i , and the output aggregates them with coefficients $\hat{f}(y)$. If f satisfies the BV promise, a single hidden unit suffices; otherwise, multiple units contribute according to $|\hat{f}(y)|^2$. Importantly, this network “growth” comes purely from superposition and measurement, without additional quantum resources.

4.2. Generalised Bernstein–Vazirani Networks

The standard BV algorithm learns within a restricted hypothesis class: the basis elements χ_y are *linear parity functions* in $\mathbb{Z}_2^n$, so the BV promise effectively assumes that f is linear. Most practical learning functions will fail this assumption, possessing dense rather than sparse Fourier spectra and requiring many samples for the approximation. Our goal in generalising the BV algorithm is to design task-adaptive basis functions (i.e., the promise) with inductive biases that better align with the target function f . We achieve this along two complementary axes:

- **Interference Operator.** In standard BVNs, the Hadamard operator induces low-expressivity parity activations. Replacing this alone directly implies alternative features. However, any alternative interference operator, being unitary, still primarily probes linear structure over $\mathbb{Z}_2^n$.
- **Expressive Input Representations.** To capture non-linearity, we enlarge the quantum system and construct non-linear representations of the input x in an auxiliary register prior to interference, making them potentially parameter-dependent to model implicit biases.

Notation. We will call a unitary I an *interference operator* if its action on computational basis states $x \in \mathbb{Z}_2^n$ induces a family of functions $\{\xi_y\}_{y \in \mathbb{Z}_2^n}$ via

$$I|x\rangle = \sum_{y \in \mathbb{Z}_2^n} \xi_y(x) |y\rangle. \quad (7)$$

The Method. We now propose a principled algorithm for *QML by interference*, inducing our *generalised BVNs*. To obtain expressive basis functions, we allocate two registers in addition to the input register: One register that encodes a

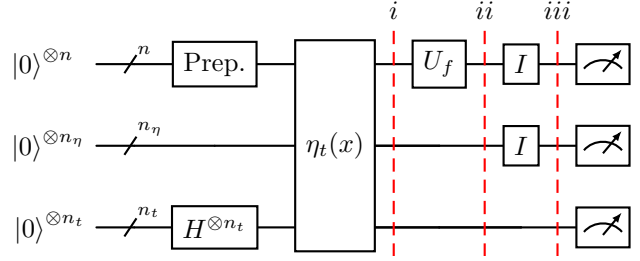


Figure 3. Our generalised Bernstein–Vazirani algorithm. We use three quantum registers: An input register, a register that computes a weighted representation of the inputs, and a weight register controlling this representation. The algorithm has three stages: (i) Prepare a uniform superposition of all inputs (Prep.) and weight configurations, and compute the representations; (ii) Query the oracle encoding the ground-truth labels across all samples in the superposition; (iii) Apply an interference operator so the labels interfere with the basis functions generated by the representation.

pre-processed *representation* $\eta_t(x) \in \mathbb{Z}_2^{n_\eta}$ of the inputs $x \in \mathbb{Z}_2^n$, and one register that stores the *weights or parameters* $t \in \mathbb{Z}_2^{n_t}$ associated with this representation.

Our generalised construction, as illustrated in Figure 3, follows the three main steps of the Bernstein–Vazirani algorithm. The inputs of the algorithm are training pairs $(x, f(x))_{i=1}^m$; a function oracle U_f to label the inputs; and an interference operator I . The main steps are as follows:

- The initial step puts all training samples and parameters of the expressive representation in uniform superposition, then runs the representation circuit:

$$|\psi_1''\rangle = \frac{1}{\sqrt{m}} \sum_x |x\rangle |0\rangle |0\rangle, \quad (8)$$

$$|\psi_1'\rangle = \frac{1}{\sqrt{m2^{n_t}}} \sum_t \sum_x |x\rangle |0\rangle |t\rangle, \text{ and} \quad (9)$$

$$|\psi_1\rangle = \frac{1}{\sqrt{m2^{n_t}}} \sum_t \sum_x |x\rangle |\eta_t(x)\rangle |t\rangle. \quad (10)$$

- The second step queries the oracle to mark the ground-truth labels $f(x)$ as amplitudes on the inputs:

$$|\psi_2\rangle = \frac{1}{\sqrt{m2^{n_t}}} \sum_t \sum_x f(x) |x\rangle |\eta_t(x)\rangle |t\rangle. \quad (11)$$

- The third step creates, from the input and representation registers, interference of the function values:

$$|\psi_3\rangle = \frac{1}{\sqrt{2^{n_t}}} \sum_{yzt} \hat{f}(yzt) |y\rangle |z\rangle |t\rangle, \quad (12)$$

where

$$\hat{f}(yzt) = \frac{1}{m} \sum_x f(x) \chi_{yzt}(x), \quad (13)$$

$$\chi_{yzt}(x) = \sqrt{m} \xi_y(x) \xi_z(\eta_t(x)). \quad (14)$$

Sampling Equation (12) yields, with probability $|\hat{f}(yzt)/\sqrt{2^{n_t}}|^2$, the dominant basis vectors $y \in \mathbb{Z}_2^{n_y}$, $z \in \mathbb{Z}_2^{n_z}$, $t \in \mathbb{Z}_2^{n_t}$, and thus the corresponding basis functions χ_{yzt} . Since the extended basis may be non-orthonormal due to its over-completeness relative to the effective number of training samples m , we must reconstruct each sampled coefficient $\hat{f}(yzt)$, which is done efficiently as described next.

Reconstructing the Coefficients. Let $X \in \mathbb{R}^{m \times k}$ denote the matrix of k basis evaluations on the training set, $X_{ij} = \chi_j(x_i)$. Let $F \in \mathbb{R}^m$ be the vector of target values on the training set, and $\hat{F} \in \mathbb{R}^k$ the vector of Fourier coefficients. We aim to recover F from the basis X , or the best approximation of f in χ . In fact, the coefficients $\hat{F}$ need to be scaled properly, as the sampling has learned the dominant correlations but not the overall scaling. A standard approach to recover the coefficients is to solve the ridge regression problem $\min_{\hat{F}} \|F - X\hat{F}\|_2^2 + \lambda \|\hat{F}\|_2^2$ via the Gram matrix:

$$G = X^\top X, \quad \hat{F} \leftarrow (G + \lambda I)^{-1} X^\top F, \quad (15)$$

for a regularisation factor $\lambda \in \mathbb{R}_+$. In practice, this reconstruction is comparable to a single training epoch in classical models, as it involves a single pass over the training set.

Making New Predictions. A new prediction on a data point x is made by evaluating

$$f(x) = \sum_{yzt} \hat{f}(yzt) \chi_{yzt}(x). \quad (16)$$

Our construction generalises the standard Bernstein–Vazirani algorithm in the sense that we sample BV basis functions $\xi_y(x)$ if $\xi_z(\eta_t(x))$ is a constant for all x, z , and t . A summarising pseudocode of our generalised BVNs is provided in Algorithm 1.

The Generalised Bernstein–Vazirani Network. The generalised BVN architecture is shown in Figure 1b. Unlike standard BVNs, the hidden units are no longer simple parity functions but expressive sub-models. Inputs x_i pass through a hidden layer of optimised components χ_{yzt} , whose predictions are aggregated at the output and weighted by the interference amplitudes $\hat{f}(yzt)$. The architecture behaves similarly to an ensemble, but the ensemble is generated implicitly through interference rather than by training multiple models. Crucially, *this rich ensemble incurs no additional quantum resource cost: only a single model is physically implemented, while the remaining effective models emerge purely through superposition and measurement.*

4.3. The Expressive Representation of Inputs

We now elaborate on our *expressive representation* η of the inputs. This representation is central to obtaining smooth,

Algorithm 1 Generalised Bernstein–Vazirani Networks (Quantum Machine Learning by Interference)

Input: Training pairs $(x, f(x))_{i=1}^m$; Function oracle U_f ; Interference operator I ; Number of shots n_{shots} .

while n_{shots} not reached **do**

 Prepare superposition of inputs x via Equation (8).

 Prepare superposition parameters t via Equation (9).

 Compute representation $\eta_t(x)$ via Equation (10).

 Query function oracle U_f via Equation (11).

 Interfere the x and η registers via Equation (12).

 Measure basis states $\{|y\rangle |z\rangle |t\rangle\}$.

end while

for Parameter configuration $(y, z, t) \in \{|y\rangle |z\rangle |t\rangle\}$ **do**

 Evaluate $\chi_{yzt}(\cdot) = \sqrt{m} \xi_y(\cdot) \xi_z(\eta_t(\cdot))$ on training set.

end for

 Compute Fourier coefficient $\hat{f}(yzt)$ via Equation (15).

Return: Fourier pairs $(\hat{f}(yzt), \chi_{yzt})$.

regularised models that generalise well.

In principle (Nielsen & Chuang, 2010, Section 1.4.1), any Boolean function on binary inputs can be computed coherently: there exists a unitary A (constructed from Toffoli and CNOT gates; and ancilla qubits) such that

$$A |x\rangle |0\rangle |t\rangle = |x\rangle |\eta_t(x)\rangle |t\rangle. \quad (17)$$

That is, as an expressive representation, we could mirror standard classical network architectures—whose effectiveness is well established—to construct η in a reversible quantum form. Appendix D discusses a quantum implementation of a classical 2-layer MLP with step activation functions and fixed biases. In practice, however, implementing A to mirror classical models requires ancilla qubits and uncomputation, and induces nontrivial circuit depths.

Constrained by the available qubits, we introduce a structured, resource-efficient rectangle representation η_{rect} . For instance, since the standard BV basis is already complete, our rectangle representations activate parameter-dependent hyper-rectangles of the input domain. This has the effect, after interference, of modifying the BV basis locally on different rectangles. Implementation details and ablations are discussed in Appendix E.

4.4. The Interference Operator

We lastly discuss three candidate interference operators: Hadamard, Fourier, and Chebyshev. The interference operator determines the hidden activation and model expressivity, e.g., Hadamard produces step-like patterns, while Fourier and Chebyshev yield smoother variations (see Appendix F).

The Hadamard Transform. The Hadamard transform H (Nielsen & Chuang, 2010) is the most natural interference

operator on $\mathbb{Z}_2^n$. It acts on a basis state $|x\rangle$ as

$$H|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_2^n} (-1)^{x \odot y} |y\rangle, \quad (18)$$

where x and y are binary vectors. The phase factor $(-1)^{x \odot y}$ encodes the parity of the binary inner product. The Hadamard transform is the Fourier transform on $\mathbb{Z}_2^n$.

The Fourier Transform. A more general Fourier transform F (Nielsen & Chuang, 2010) arises when we embed $\mathbb{Z}_2^n$ into the complex vector space indexed by integers in $\mathbb{Z}_{2^n}$. In this viewpoint, the x is interpreted as an integer in $\{0, \dots, 2^n - 1\}$. The transform acts on basis states as

$$F|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_2^n} e^{i2\pi xy/2^n} |y\rangle, \quad (19)$$

where xy denotes standard integer multiplication.

The Chebyshev Transform. The Chebyshev transform T (Williams et al., 2023) constructs Chebyshev polynomials on discretised Chebyshev nodes. For continuous inputs $x \in [-1, 1]$, Chebyshev polynomials are defined as $\chi_y(x) = \cos(y \arccos(x))$. In the discrete setting, we view $x \in \mathbb{Z}_2^n$ as an integer and map it to a Chebyshev node $x^{\text{ch}} := \cos(\pi(2x + 1)/2^{n+1})$. Defining polynomials $T_y(x) := \cos(y \arccos(x^{\text{ch}}))$, the transform acts as

$$T|x\rangle = \frac{1}{2^{\frac{n}{2}}} T_0(x) |0\rangle + \frac{1}{2^{\frac{n-1}{2}}} \sum_{y \in \mathbb{Z}_2^n \setminus \{0\}} T_y(x) |y\rangle. \quad (20)$$

5. Experimental Results

We validate our BVNs on several representative supervised learning tasks of adequate size for classical simulation, from hand-crafted sanity-check tasks Section 5.1), through synthetic 2D and real-world 4D classification (Section 5.2), to implicit image representation (Section 5.3). Section 6 analyses complexities and compares qubits, parameters, shots and runtime costs, with extended discussions covering the sampling complexity in Appendix G. Section 7 lastly performs ablation studies on the proposed BVNs. All quantum circuits, unless otherwise stated, are simulated noise-free using PennyLane’s `lightning.qubit` (Bergholm et al., 2018) to investigate properties of the new paradigm. We compute on a conventional computer (AMD Ryzen 9 5900X 12-Core Processor CPU with 128GB RAM). The classification metric is the top-1 accuracy between ground-truths and rounded predictions.

5.1. Fitting the Expressive Representation in 2D

In Figure 1c we demonstrate the efficiency of the generalised model over the standard one, constructing 2D classification

tasks whose regions match our expressive basis functions, making the target directly representable in the generalised basis. With Chebyshev interference and only 100 shots, the generalised BVN achieves 100% accuracy, whereas the standard BVN reaches only $\sim 70\%$.

5.2. Classification in 2D and 4D

We experiment with classification tasks. First, we solve binary classification in 2D on a variety of synthetic datasets shown in Figure 4a, including the canonical blobs, moons, and a circle, as well as several hand-made shapes with different boundaries, yielding ten datasets S1-S10. Second, we classify the real-world Iris (Fisher, 1936) (150 samples) and Penguins (Horst et al., 2022) datasets (333 samples), each consisting of 4D vectors across three classes (1,2,3). Data are normalised to the qubit grid. We benchmark our BVNs with Hadamard and Chebyshev operators against a classical 3-layer MLP baseline with 10 units per layer, a classical SVM model, and a 20-layer universal quantum single-qubit classifier (SQC) (Pérez-Salinas et al., 2020). We vary the training set size (25%, 50%, 75%, 100%) and evaluate on the full datasets to assess how well the models *generalise*, i.e., learn the classification boundaries. Appendix H inspects sampled pairs $(\hat{f}, \chi)$. Results report the mean performance over five runs per model and dataset.

Figure 4b presents the binary classification results in 2D. Our BVNs successfully learn the decision boundaries. With only 50% of the training data, they reach nearly 100% accuracy. In the low training split, we observe that increasing the regularisation parameter λ improves generalisation. The MLP, the SVM and single-qubit classifier baselines struggle on datasets with high variation (S2, S6, S9), whereas the BVN bases are expressive enough to capture high variations. The MLP’s behaviour reflects its spectral bias, and we conjecture that the SQC is constrained by its depth.

Figure 5 shows the 4D results. Here, we observed that the small number of training samples (< 500) relative to the large Hilbert-space dimension (2^{16}) leads to severe sampling leakage (Wakeham & Schuld, 2024). Assigning a dummy non-zero fill-label to inputs outside the training set mitigated this issue. We use $\lambda = 0.1$, refer to Appendix I for other values of λ and to Section 7.1 for Ablations on fill fraction and value. Unfilled data lead to poor generalisation, whereas filled data, while degrading the standard BVN, substantially improve the generalised BVN, bringing it on par with baselines. We attribute the SVM’s slightly better performance over the MLP to the small size of the datasets.

5.3. Implicit Image Representation

We run BVNs with $\lambda = 0.1$ and Chebyshev on implicit image representations. We benchmark against the classical MLP+RFF and SIREN models (Benbarka et al., 2022),

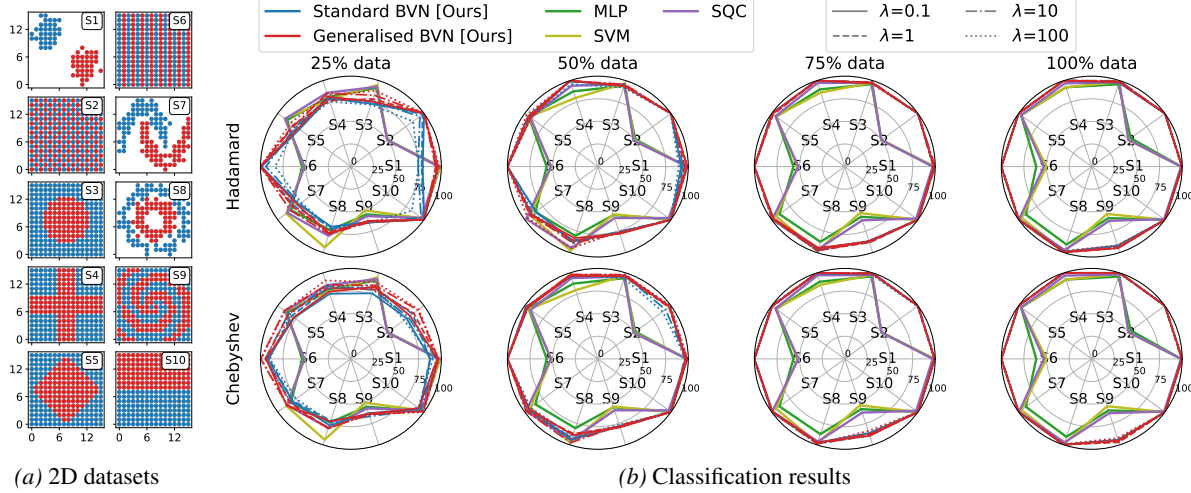


Figure 4. Binary classification results on synthetic 2D datasets. The top and bottom rows show BVN results with Hadamard and Chebyshev interference, with columns specifying data fractions for training. Chebyshev outperforms Hadamard; the generalised BVN outperforms the standard BVN. BVNs outperform MLP, SVM, and SQC, particularly on high-frequency shapes (S2, S6, and S9).

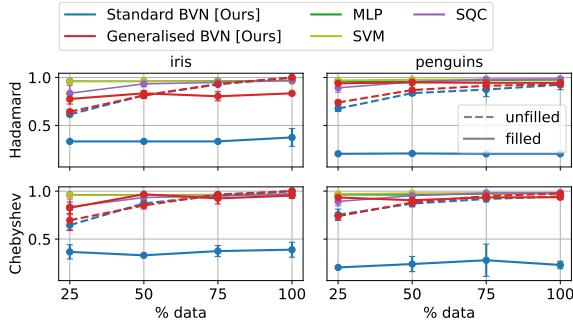


Figure 5. Real-world classification results on Iris and Penguins datasets, with filled vs. unfilled (original) comparison for BVNs. Filled data substantially improves the generalised BVN, allowing $>90\%$ accuracy with only 25% data, comparable to the baselines.

and the quantum QIREN (Zhao et al., 2024) and QVF (Wang et al., 2025) models. The models are trained on a coarse 64×64 image, evaluated on a finer 128×128 grid to assess interpolation, and we report PSNR and MSE. We use 10,000 shots to demonstrate that BVNs can represent high-frequency signals. Ablations in Figures 10 and 13 show that even 1,000 shots yield good results visually. Figure 6 presents our results. Again, the generalised BVN outperforms the standard one and stands competitively alongside the baselines. While it does not always yield the highest PSNR, it is visually more regular and coherent.

6. Computational Complexity

Our labelled dataset encoding in BVNs has a cost of $O(mn)$ for m training samples and n qubits. The rectangle representations with input and parameter resolutions n and n_t yield $O(nn_t)$ gates. The Hadamard (Nielsen & Chuang, 2010) or Chebyshev (Williams et al., 2023) operator costs $O(n)$ or $O(n + n^2)$ gates. Coefficient reconstruction scales as $O(mk(c + k + 1) + k^3)$ for k sampled states. We provide

a full discussion in Appendix G.

The total data-loading cost of BVNs, given by $O(mn) \times n_{\text{shots}}$, scales significantly more favorably than that of PQC models, which is $O(mn) \times (2c \cdot p \cdot e)$, where $2c$ denotes the number of shots used for each partial derivative under the parameter-shift rule, p is the number of trainable parameters, and e is the number of training epochs. Table 2 provides a systematic resource comparison between BVNs and benchmarks, including the estimated shot count of parameter-shift training and true simulation runtime using back-propagation. BVNs are markedly efficient: our 8-22-qubit BVNs are 2-3 orders of magnitude faster than the SQC or the multi-qubit QIREN and QVF.

Approximation Error. Lastly, Section G.2 bounds the approximation error of BVNs. Let $\Omega \subset \mathbb{Z}_2^n$ be a sampled subset of basis states and f_Ω the truncated approximation of f using basis states in Ω . The approximation error $\epsilon = \mathbb{E}_x[(f(x) - f_\Omega(x))^2]$ can be written as $\epsilon = \hat{F}_\Omega^\top G_\Omega \hat{F}_\Omega$, with $\hat{F}_\Omega = (\hat{f}(a))_{a \notin \Omega}$ denoting the residual coefficients and G_Ω the Gram matrix from Equation (15). Since G_Ω is positive semidefinite, we have $\epsilon \leq \lambda_{\max}(G_\Omega) \|\hat{F}_\Omega\|_2^2$, where $\lambda_{\max}(G_\Omega)$ is its largest eigenvalue.

For standard BVNs, we have $\|\hat{F}_{\mathbb{Z}_2^n}\|_2 = 1$ and $G = I$, yielding $\epsilon = \|\hat{F}_\Omega\|_2^2 = 1 - \|\hat{F}_\Omega\|_2^2$. Thus, the approximation error is determined entirely by the unsampled spectrum. For generalised BVNs, the induced basis is generally overcomplete and non-orthogonal, yielding $\epsilon \leq \lambda_{\max}(G_\Omega) \|\hat{F}_\Omega\|_2^2$, showing a trade-off between the non-orthogonality of the extended basis and the expressive representation: Stronger correlations between basis functions increase the penalty factor $\lambda_{\max}(G_\Omega)$, while the richer representation η_t can yield expressive representations of the target that substantially reduce the residual norm $\|\hat{F}_\Omega\|_2^2$.

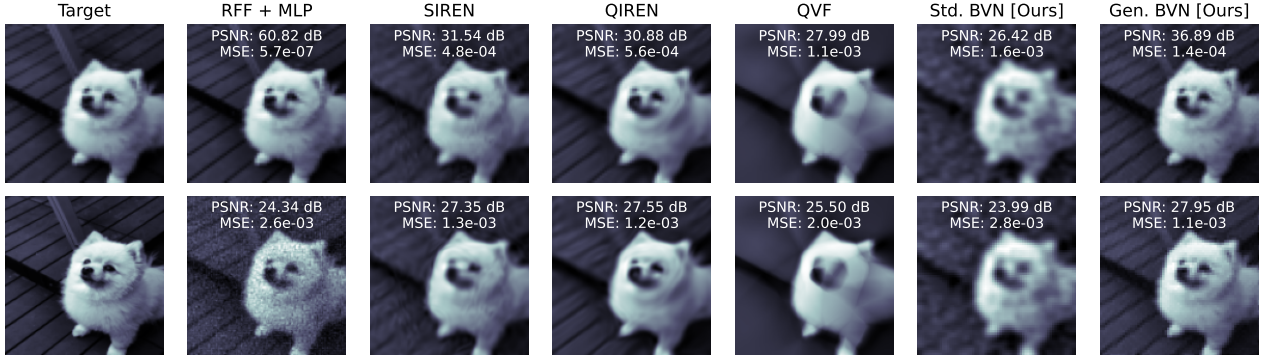


Figure 6. Implicit image representation. The top row shows results on the coarse (64×64) training image, while the bottom row shows evaluations on a finer (128×128) test image. The generalised BVN outperforms the standard BVN and is competitive with the benchmarks.

Table 2. Resource comparison across all models (quantum methods in blue). We report the expected shot count for parameter-shift training of PQCs and the runtime using back-propagation. Our BVNs are significantly more efficient in terms of shots and runtime.

Model	Num. of qubits	Params (PQC)	Params (class.)	Epochs ≡ Basis states for BVNs	Shots (parameter-shift)	Runtime (back-prob.)
Classification (2D), Figure 4						
MLP	–	–	151	30	–	2.70s
SVM	–	–	253	–	–	0.001s
SQC	1	120	–	30	120 · 30(2c)	37.5s
Std. BVN	8(+1)	–	–	45	100	0.027s
Gen. BVN	14(+1)	–	–	95	100	0.039s
Classification (4D), Figure 5						
MLP	–	–	171	100	–	18.34s
SVM	–	–	178	–	–	0.0015s
SQC	1	240	–	30	240 · 30(2c)	819.35s
Std. BVN	16(+1)	–	–	95	100	1.36s
Gen. BVN	22(+1)	–	–	100	100	4.65s
Image Fitting, Figure 6						
RFF+MLP	–	–	68,289	100	–	1.8s
SIREN	–	–	24,073	100	–	1.2s
QIREN	6	270	899	2,000	270 · 2000(2c)	413.8s
QVF	4	80	152,848	2,000	80 · 2000(2c)	142.0s
Std. BVN	12(+1)	–	–	288	10,000	4.8s
Gen. BVN	24(+1)	–	–	6,383	10,000	82.5s

7. Ablations

7.1. Dummy Filling

We investigate the sensitivity of the generalised BVN to the dummy-fill value and fraction. We vary both and report results in Table 3 using tuples (value, fraction), with (0, 0) denoting unfilled data and (4, 1) the setting used in Figure 5.

Benchmark methods are omitted as they fail on the filled data due to severe class imbalance: the $m < 500$ informative samples are expanded to 2^{16} points, with the remaining points assigned a constant dummy label. In contrast, the generalised BVN remains performant across a wide range of fill values and fractions.

7.2. Random Sampling

To isolate the value of quantum interference, we keep the coefficient reconstruction step from Equation (15) unchanged

Table 3. Influence of the dummy filling on the performance of BVNs. Tuples (V, F) indicate dummy fill (value, fraction).

		(0, 0.0)	(4, 0.25)	(4, 0.5)	(4, 0.75)	(4, 1)	(10, 1)	(100, 1)	(1000, 1)
Iris	Std. BVN	0.84	0.78	0.65	0.39	0.33	0.33	0.33	0.33
	Gen. BVN	0.77	0.76	0.74	0.99	0.88	0.99	0.93	0.85
Penguins	Std. BVN	0.60	0.38	0.40	0.23	0.20	0.20	0.20	0.20
	Gen. BVN	0.59	0.86	0.85	0.87	0.96	0.93	0.95	0.96

and replace the quantum-selected basis functions with uniformly random basis functions of identical cardinality.

Expected Coverage. For an n -qubit input and an $(n_\eta + n_t)$ -qubit extended register for expressive representation, the total measurement space dimension is $2^{n+n_\eta+n_t}$, but the relevant diversity for the input features is determined by the marginal distribution over the n -bit input register: $p(y) = \sum_{z,t} p(y, z, t)$. For uniform sampling, $(y, z, t) \sim \mathcal{U}(2^{n+n_\eta+n_t})$, the probability of observing y is $p(y) =$

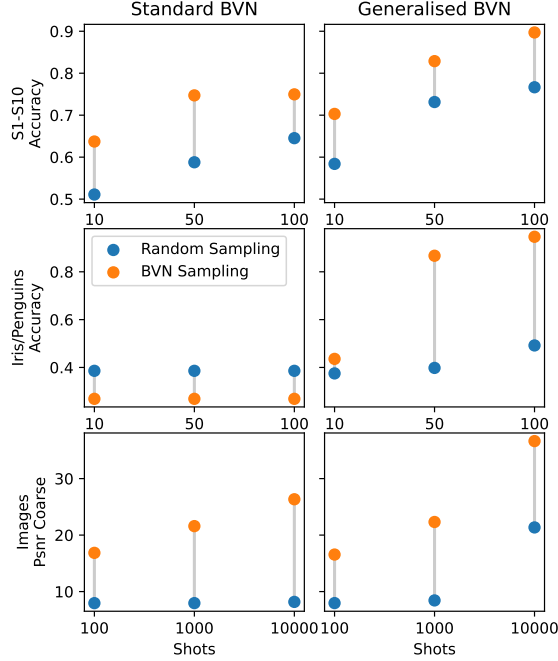


Figure 7. Ablation of interference-based sampling versus random uniform sampling. BVN’s performance indicates that interference effectively identifies informative basis functions.

$2^{n_\eta+n_t}/2^{n+n_\eta+n_t} = 1/2^n$, showing that the additional qubits do not improve the marginal probability of sampling a distinct input feature y . Yet, the expected coverage, or number of unique y , after N random draws, is

$$E = (\text{number of } y) \cdot (1 - \text{prob}(y \text{ never occurs})) \quad (21)$$

$$= 2^n(1 - (1 - 1/2^n)^N) \approx 2^n(1 - e^{-N/2^n}), \quad (22)$$

which shows that the number of shots for covering a constant fraction of the input space scales exponentially with n .

Empirical Evidence. In Figure 7, we validate the above analysis by comparing the performance of the approximated function using basis functions identified by our BVNs and uniform random sampling. We report average classification accuracy across the synthetic shapes in Figure 4, the real-world filled Iris and Penguins datasets in Figure 5, and PSNRs from the image fitting experiment in Figure 6.

BVNs consistently outperform random sampling. As expected, the advantage is amplified as the search space grows: compare synthetic and real-world results using $N = 100$ shots for $n = 8/14$ and $n = 16/22$ qubits in the standard/generalised BVN, respectively. For the image fitting experiment ($n = 12$), random sampling benefits from a larger measurement budget at 10^4 shots, reaching ~ 23 dB PSNR. Yet, both BVNs rapidly identify informative basis functions, particularly in the low-shot regimes ($N = 100, 1000$).

Table 4. Accuracy and runtime comparison in noise simulation. The numbers in brackets are in seconds. The experimental setting does not apply for classical methods (denoted by “-”).

	Noise-free	Noise	Random
Std. BVN	0.99 (0.005)	0.86 (13.610)	0.80 (0.005)
Gen. BVN	0.98 (0.028)	0.92 (13.525)	0.85 (0.028)
SQC	0.96 (112.066)	0.99 (11513.526)	-
MLP	0.94 (3.801)	-	-
SVM	0.98 (0.010)	-	-

7.3. Robustness to Noise

To evaluate robustness to hardware noise, we compare BVNs and SQC (Pérez-Salinas et al., 2020) on the S8 shape of Figure 4a using a NISQ-inspired noise model (AbuGhanem, 2026): Depolarizing errors of 5×10^{-3} on 1-qubit gates, 5×10^{-2} on 2-qubit gates, and a 2% readout bit-flip error. State preparation in BVNs is decomposed into elementary gates to expose this step to noise. SQC is trained using backpropagation without noise and parameter-shift with noise. We report accuracy and runtime for both settings and compare BVNs against random selection from Section 7.2. As shown in Table 4, BVNs remain noise-robust and outperform random selection. SQC performs the best, but requires ~ 3.2 hours of parameter-shift training, while BVNs take < 15 seconds at competitive accuracy.

8. Practical Recipe for BVNs

A central design question in BVNs is how to choose the interference operator, representation, and parameter resolution for a given application. The key principle is to select them such that the induced basis aligns with the target function:

- We replace the standard BV basis $\xi_y(x) = (-1)^{x \odot y}$ with $\chi_{yzt}(x) = \sqrt{m}\xi_y(x)\xi_z(\eta_t(x))$ to obtain potentially sparser representations. For example, a function requiring two BV basis states in ξ may become one-sparse in χ under a suitable representation.
- Beyond sparsification, generalised BVNs enable spectrum reshaping. For example, a standard representation with coefficients $\sqrt{5/100}$ and $\sqrt{95/100}$ makes the smaller informative component unlikely to be sampled. A suitable representation can redistribute this to make informative components more accessible.

Ultimately, these design choices should be guided by assumed structures in the data, like convolutional architectures for images or attention mechanisms for sequences. A practical heuristic is to choose an interference operator expected to concentrate the target spectrum, a representation that encodes task-specific structures or invariances, and vary the parameter resolution to control the resulting expressivity.

9. Conclusion

We proposed an interference-based alternative to variational QML by reinterpreting the Bernstein–Vazirani algorithm as a neural model, yielding Bernstein–Vazirani Networks (BVNs). Our generalised BVNs incorporate inductive bias to improve fitting and reduce measurement cost. Experimentally, the generalised BVNs achieved nearly 100% classification accuracy using only 50% of the training data, and reached close to 40 dB PSNR on image-fitting, consistently surpassing the standard model and often outperforming quantum and classical baselines. Our work advances QML by introducing a conceptually new framework, which, we believe, will lead to a plethora of improvements.

Limitations and Future Work. While our BVNs make a first step in a fundamentally new QML direction, many open questions remain. For instance, while the generalised BVN performs well on synthetic data, scaling to real-world data (e.g., Iris, Penguins) features spectral leakage. Beyond our ad-hoc dummy-fill, basis engineering to align the parameter manifold with the data could help. Also, rectangle representations, while expressive, introduce artefacts and may hurt interpolation. Future work would explore smoother alternatives. We detail these directions in Appendix J.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project number 534951134. T. Birdal was supported by a UKRI Future Leaders Fellowship [grant number MR/Y018818/1]. NKM and MM acknowledge support from the Lamarr Institute for Machine Learning and Artificial Intelligence.

References

- AbuGhanem, M. Early ibm quantum computers: architectural analysis and performance benchmarks. *The Journal of Supercomputing*, 82(8):422, 2026.
- Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., and Wolf, R. Training deep quantum neural networks. *Nature communications*, 11(1):808, 2020.
- Benbarka, N., Höfer, T., Zell, A., et al. Seeing implicit neural representations as fourier series. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2041–2050, 2022.
- Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M. S., Alonso-Linaje, G., Akash-Narayanan, B., Asadi, A., et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.
- Bernstein, E. and Vazirani, U. Quantum complexity theory. In *ACM symposium on Theory of computing*, pp. 11–20, 1993.
- Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.
- Cao, Y., Guerreschi, G. G., and Aspuru-Guzik, A. Quantum neuron: an elementary building block for machine learning on quantum computers. *arXiv preprint arXiv:1711.11240*, 2017.
- Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L., and Coles, P. J. Challenges and opportunities in quantum machine learning. *Nature computational science*, 2(9):567–576, 2022.
- Cong, I., Choi, S., and Lukin, M. D. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.
- da Silva, A. J., Luderer, T. B., and de Oliveira, W. R. Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks*, 76:55–64, 2016.
- Deutsch, D. and Jozsa, R. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.
- Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.
- Gleinig, N. and Hoefler, T. An efficient algorithm for sparse quantum state preparation. In *Design Automation Conference (DAC)*, pp. 433–438. IEEE, 2021.
- Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Herman, D., Raymond, R., Li, M., Robles, N., Mezzacapo, A., and Pistoia, M. Expressivity of variational quantum machine learning on the boolean cube. *Quantum Engineering*, 4:1–18, 2023.
- Horst, A. M., Hill, A. P., and Gorman, K. B. Palmer archipelago penguins data in the palmerpenguins r package-an alternative to anderson’s irises. *R Journal*, 14(1), 2022.
- Kapoor, A., Wiebe, N., and Svore, K. Quantum perceptron models. *Advances in neural information processing systems*, 29, 2016.

- Larocca, M., Thanasilp, S., Wang, S., Sharma, K., Biamonte, J., Coles, P. J., Cincio, L., McClean, J. R., Holmes, Z., and Cerezo, M. Barren plateaus in variational quantum computing. *Nature Reviews Physics*, pp. 1–16, 2025.
- Liu, W., Zhu, Y., Zha, Y., Wu, Q., Jian, L., and Liu, Z. Rotation-and permutation-equivariant quantum graph neural network for 3d graph data. *Transactions on Pattern Analysis and Machine Intelligence*, 2025.
- McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9 (1):4812, 2018.
- Meli, N. K., Wang, S., Benkner, M. S., Sasdelli, M., Chin, T.-J., Birdal, T., Moeller, M., and Golyanik, V. Quantum-enhanced computer vision: Going beyond classical algorithms. *arXiv preprint arXiv:2510.07317*, 2025.
- Mozafari, F., De Micheli, G., and Yang, Y. Efficient deterministic preparation of quantum states using decision diagrams. *Physical Review A*, 106(2):022617, 2022.
- Nguyen, Q. T., Schatzki, L., Braccia, P., Ragone, M., Coles, P. J., Sauvage, F., Larocca, M., and Cerezo, M. Theory for equivariant quantum neural networks. *PRX Quantum*, 5(2):020328, 2024.
- Nielsen, M. A. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.
- Nielsen, M. A. and Chuang, I. L. *Quantum computation and quantum information*. Cambridge university press, 2010.
- O’Donnell, R. *Analysis of boolean functions*. Cambridge University Press, 2014.
- Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., and Latorre, J. I. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.
- Ramacciotti, D., Lefterovici, A. I., and Rotundo, A. F. Simple quantum algorithm to efficiently prepare sparse states. *Physical Review A*, 110(3):032609, 2024.
- Recio-Armengol, E., Ahmed, S., and Bowles, J. Train on classical, deploy on quantum: scaling generative quantum machine learning to a thousand qubits. *arXiv preprint arXiv:2503.02934*, 2025.
- Schatzki, L., Larocca, M., Nguyen, Q. T., Sauvage, F., and Cerezo, M. Theoretical guarantees for permutation-equivariant quantum neural networks. *npj Quantum Information*, 10(1):12, 2024.
- Schuld, M. and Petruccione, F. Supervised learning with quantum computers. *Quantum science and technology*, 17, 2018.
- Schuld, M., Sinayskiy, I., and Petruccione, F. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.
- Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., and Killo-ran, N. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.
- Schuld, M., Sweke, R., and Meyer, J. J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3): 032430, 2021.
- Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
- Sim, S., Johnson, P. D., and Aspuru-Guzik, A. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.
- Wakeham, D. and Schuld, M. Inference, interference and invariance: How the quantum fourier transform can help to learn from data. *arXiv preprint arXiv:2409.00172*, 2024.
- Wang, S., Theobalt, C., and Golyanik, V. Quantum visual fields with neural amplitude encoding. In *39th Annual Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2025.
- Williams, C. A., Paine, A. E., Wu, H.-Y., Elfving, V. E., and Kyriienko, O. Quantum chebyshev transform: Mapping, embedding, learning and sampling distributions. *arXiv preprint arXiv:2306.17026*, 2023.
- Zhao, J., Qiao, W., Zhang, P., and Gao, H. Quantum implicit neural representations. *arXiv preprint arXiv:2406.03873*, 2024.

Appendix

This appendix entails further technical details and experiments to complement the main text. It includes:

Title	Text	Appendix
Derivation of the interference behind the Bernstein-Vazirani algorithm	Section 4.1	Appendix A
Proof of the Fourier Expansion Theorem	Theorem 4.1	Appendix B
Derivation of BVNs from a Linear Algebra Perspective	Theorem 4.1	Appendix C
Layered Expressive Representation	Section 4.3	Appendix D
Rectangle Expressive Representation	Section 4.3	Appendix E
Interference Operators in Comparison	Section 4.4	Appendix F
Complexities and Systematic Comparison	Section 5	Appendix G
Inspecting the sampled Coefficients	Section 5.2	Appendix H
Comparing Filled and Unfilled Training Data	Section 5.2	Appendix I
Future Work and Technical Extensions	Section 9	Appendix J

A. Interference in Bernstein–Vazirani

We shortly review the Bernstein–Vazirani algorithm (Bernstein & Vazirani, 1993) and explain how interference occurs.

From the three steps of the Bernstein–Vazirani algorithm (Equations (2) to (4)) and after replacing f by its signature $f(x) = s \odot x$, we have

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_n^2} |x\rangle \quad (23)$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_n^2} (-1)^{f(x)} |x\rangle \quad (24)$$

$$|\psi_3\rangle = \sum_{y \in \mathbb{Z}_n^2} \hat{f}(y) |y\rangle. \quad (25)$$

with Fourier coefficients

$$\hat{f}(y) = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_n^2} (-1)^{s \odot x + y \odot x} \quad (26)$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_n^2} (-1)^{(s \oplus y) \odot x} \text{ mod } 2. \quad (27)$$

We want to show that $\hat{f}(y) = 1$ if and only if $y = s$ and 0 otherwise. On $\mathbb{Z}_n^2$, we use two important properties:

$$(a + b) \text{ mod } 2 = (a \text{ mod } 2 + b \text{ mod } 2) \text{ mod } 2, \quad (28)$$

$$(a \cdot b) \text{ mod } 2 = (a \text{ mod } 2) \cdot (b \text{ mod } 2) \text{ mod } 2. \quad (29)$$

After applying Equations (28) and (29) to the phase term in

Equation (27), it holds

$$(s \odot x + y \odot x) \text{ mod } 2 \quad (30)$$

$$= (s^\top x \text{ mod } 2 + y^\top x \text{ mod } 2) \text{ mod } 2 \quad (31)$$

$$= ((s \text{ mod } 2)^\top (x \text{ mod } 2) + (y \text{ mod } 2)^\top (x \text{ mod } 2)) \text{ mod } 2 \quad (32)$$

$$= ((s \text{ mod } 2 + y \text{ mod } 2)^\top (x \text{ mod } 2)) \text{ mod } 2 \quad (33)$$

$$= (((s + y) \text{ mod } 2)^\top (x \text{ mod } 2)) \text{ mod } 2 \quad (34)$$

$$= ((s \oplus y)^\top x) \text{ mod } 2 \quad (35)$$

$$= (s \oplus y)^\top \odot x. \quad (36)$$

Now substituting Equation (36) in Equation (27), we have

$$\begin{aligned} \frac{1}{2^n} \sum_{x \in \mathbb{Z}_n^2} (-1)^{(s \odot x + y \odot x) \text{ mod } 2} \\ = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_n^2} (-1)^{(s \oplus y)^\top \odot x} \end{aligned} \quad (37)$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_n^2} \prod_{i=1}^n (-1)^{(s_i \oplus y_i) \cdot x_i} \quad (38)$$

$$= \frac{1}{2^n} \prod_{i=1}^n \sum_{x_i \in \mathbb{Z}_1^2} (-1)^{(s_i \oplus y_i) \cdot x_i} \quad (39)$$

$$= \frac{1}{2^n} \prod_{i=1}^n (1 + (-1)^{(s_i \oplus y_i)}). \quad (40)$$

In the last term, we see that the complete sum evaluates to 0 if $s_i \oplus y_i = 1$ for any index i , and to 1 if $s_i \oplus y_i = 0$ for all i . On the other hand, we have $s_i \oplus y_i = 0$ if and only if $s_i = y_i$.

This collapsing sum to either 0 or 1 is called *interference*, where undesired y interfere destructively, while the desired solution interferes constructively.

B. Proof of the Fourier Expansion Theorem

In this section, we provide the proof of the Fourier Expansion Theorem 4.1.

Proof. The proof follows from linear algebra by expressing f uniquely in an orthonormal basis via inner products.

First, we verify that $\{\chi_y\}_{y \in \mathbb{Z}_n^2}$ forms an orthonormal basis of the vector space $\mathcal{F} := \{f : \{0, 1\}^n \rightarrow \mathbb{R}\}$ of functions from $\mathbb{Z}_n^2 = \{0, 1\}^n$ to $\mathbb{R}$. This is principally possible because any function in $\mathcal{F}$ can be replaced by a vector in $\mathbb{R}^{2^n}$ by enumerating all function values. We define the inner product on $\mathbb{Z}_n^2$ as $\langle f, g \rangle = \mathbb{E}_{x \in \mathbb{Z}_n^2} [f(x) g(x)]$.

As derived in Equations (37) to (40), $\forall y_1, y_2 \in \mathbb{Z}_2^n$, it holds

$$\langle \chi_{y_1}, \chi_{y_2} \rangle = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(y_1 \odot x + y_2 \odot x) \bmod 2} \quad (41)$$

$$= \begin{cases} 1, & \text{if } y_1 = y_2, \\ 0, & \text{otherwise.} \end{cases} \quad (42)$$

Since there are 2^n functions χ_y , matching the dimension of $\mathcal{F}$, the set $\{\chi_y\}_{y \in \mathbb{Z}_2^n}$ forms an orthonormal basis.

Hence, any $f \in \mathcal{F}$ expands uniquely as

$$f(x) = \sum_{y \in \mathbb{Z}_2^n} \hat{f}(y) \chi_y(x), \quad (43)$$

where

$$\hat{f}(y) = \langle f, \chi_y \rangle = \frac{1}{2^n} \sum_{z \in \mathbb{Z}_2^n} f(z) \chi_y(z). \quad (44)$$

□

C. Alternative Derivation of (Generalised) Bernstein–Vazirani Networks from a Linear Algebraic Perspective

Related, but not strictly equivalent concepts to our generalised Bernstein–Vazirani algorithm are *generalised Fourier series* or *Sparse dictionary learning*.

To (informally) generalise Theorem 4.1, we assume that there is a set of functions $\mathcal{G} = \{g_y\}_{y \in \mathbb{Z}_2^m} \subset \mathcal{F}$, such that

$$f(x) \approx \sum_{y \in \mathbb{Z}_2^m} \langle f, g_y \rangle g_y, \quad (45)$$

with $\langle \cdot, \cdot \rangle$, defined as $\langle f, g_y \rangle = \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} f(x) g_y(x)$, being the standard inner product in $\mathbb{Z}_2^n$. We further still assume access to the function oracle U_f , which realises the state

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} f(x) |x\rangle. \quad (46)$$

As our inputs x live in $\mathbb{R}^{2^n}$ and we seek expressive models, we are interested in settings where $m \geq n$, i.e., where $\mathcal{G}$ is an *overcomplete basis* for the function space $\mathcal{F}$.

Now, independently, let $\mathcal{Y} := \{|y\rangle, y \in \mathbb{Z}_2^m\}$ be some orthonormal basis, with $|y\rangle$ written in the computational basis as

$$|y\rangle = \frac{1}{\sqrt{2^m}} \sum_{x \in \mathbb{Z}_2^n} \sum_{z \in \mathbb{Z}_2^{m-n}} a_y^{x,z} |x, z\rangle, \quad (47)$$

for some amplitudes $a_y^{x,z} \in \mathbb{C}$, where the $|z\rangle$ register is an ancillary register with $(m - n)$ qubits. We embed $|\psi\rangle$

from Equation (46) in the 2^m -dimensional Hilbert space as $|\tilde{\psi}\rangle = |\psi, 0\rangle$. Expressing $|\tilde{\psi}\rangle$ in the basis $\mathcal{Y}$ yields

$$|\tilde{\psi}\rangle = \sum_{y \in \mathbb{Z}_2^m} \langle y | \tilde{\psi} \rangle |y\rangle \quad (48)$$

$$= \frac{1}{2^n} \sum_{y \in \mathbb{Z}_2^m} \sum_{p, q \in \mathbb{Z}_2^n} a_y^{p,0} f(q) \langle p | q \rangle |y\rangle \quad (49)$$

$$= \frac{1}{2^n} \sum_{y \in \mathbb{Z}_2^m} \sum_{x \in \mathbb{Z}_2^n} a_y^{x,0} f(x) |y\rangle. \quad (50)$$

Now, to connect $\mathcal{Y}$ and $\mathcal{G}$, we observe that the amplitudes $a_y^{x,0}$ define the family of functions

$$g_y(x) := a_y^{x,0}. \quad (51)$$

Even though the initial state has ancilla qubits in $|0\rangle$, the *different* y basis vectors spread across the ancillary dimensions, giving rise to an *overcomplete dictionary* of functions g_y . Constructing the basis $\mathcal{Y}$ such that $g_y(x) = a_y^{x,0}$ transforms Equation (50) into

$$|\tilde{\psi}\rangle = \sum_{y \in \mathbb{Z}_2^m} \langle g_y, f \rangle |y\rangle. \quad (52)$$

Measuring $|\tilde{\psi}\rangle$ in the basis $\mathcal{Y}$ reveals, with probability $|\langle g_y, f \rangle|^2$, the corresponding y and the associated g_y needed to approximate f according to Equation (45).

If $n = m$ and the set $\mathcal{G}$ is orthonormal in $\mathcal{F}$, we recover the standard BV setting (up to the chosen basis $\mathcal{Y}$). The generalised BV setting emerges when $m \geq n$, giving rise to an overcomplete and expressive basis.

How to design $\mathcal{Y}$ that realises $\mathcal{G}$? This is now an architectural design choice. The important point is that enlarging the system with ancillary qubits allows us to realise an overcomplete basis. The interference operator acting on the representation register effectively disentangles it from the input x , allowing us to write Equation (50) as a sum over x while isolating the $|y\rangle$ register.

How to measure in general bases? We are interested in measuring $|\tilde{\psi}\rangle$ in a basis $\mathcal{Y}$ that is not necessarily the computational basis. Let Y be the measurement observable of the basis $\mathcal{Y}$. As a Hermitian operator, Y can be diagonalised. There exists a unitary U_Y such that

$$U_Y Y U_Y^\dagger = D, \quad (53)$$

where D is diagonal in the computational basis. Then,

$$\langle \tilde{\psi} | Y | \tilde{\psi} \rangle = \langle \tilde{\psi} | U_Y^\dagger D U_Y | \tilde{\psi} \rangle. \quad (54)$$

Measuring $|\tilde{\psi}\rangle$ in the basis $\mathcal{Y}$ thus reduces to implementing U_Y followed by measurement in the computational basis. This is exactly the action of our expressive representation circuit followed by the interference operator.

D. Layered Expressive Representation

In the following, we provide a quantum implementation of a classical two-layer MLP with step activation functions and fixed biases as an exemplary expressive representation to be used in our generalised BVNs (see Section 4.3).

Equation (17) is written compactly for the input, activation, and weights. In practice, the system may be organised into several sub-registers to support different input dimensions and to implement a classical layered representation reversibly.

Definition D.1 (Quantum Layer). A quantum layer ℓ consists of multiple units. Each unit is specified by an output register $|0\rangle$, an activation function σ , and d weight registers $|t\rangle$, where d is the dimension of the activation vector of layer $\ell - 1$. Given an input activation register $|\sigma_{\ell-1}\rangle$, the layer implements the coherent update

$$|\sigma_{\ell-1}\rangle |0\rangle |t\rangle \mapsto |\sigma_{\ell-1}\rangle |\sigma_\ell\rangle |t\rangle, \quad \sigma_\ell = \sigma\left(\sum_{i=1}^d t_i (\sigma_{\ell-1})_i\right). \quad (55)$$

Ancillary qubits are abstracted but may be required for reversible arithmetic.

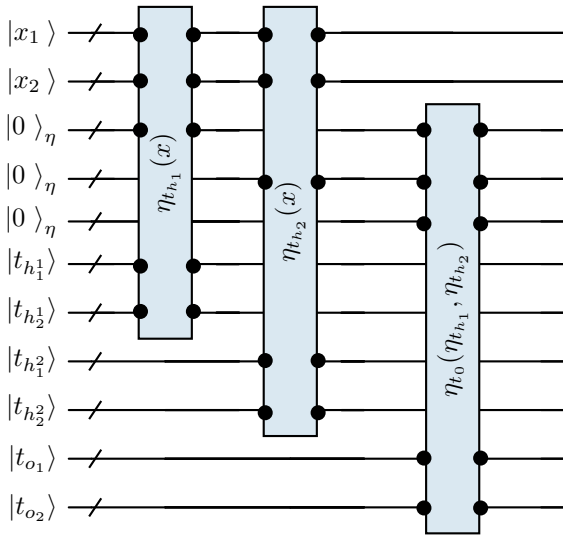


Figure 8. A quantum circuit realising an expressive representation of the input. It implements a fully connected two-layer network. The input is a two-dimensional register $|x_1\rangle |x_2\rangle$. The hidden layer consists of two units, each computing $\eta_{t_{h_j}}(x) = \sigma(t_{h_j^1} x_1 + t_{h_j^2} x_2)$ and the output unit computes η_o in the same way, but acting on the hidden activations instead of the input. The circuit is implemented using quantum arithmetic with ancillary qubits (not shown).

In Figure 8, we illustrate a quantum circuit implementing an expressive representation of the input. It realises a fully

connected two-layer network. The 2-dimensional input $|x\rangle$ is stored in two sub-registers, forming the input layer. The hidden layer contains two units, each with two weight registers that modulate the input; the weighted sum is passed through an activation and written to an activation register. The output layer is a single unit acting on the hidden activations. For this example, the decomposition $|x\rangle |\eta_t(x)\rangle |t\rangle$ from Equation (17) becomes

$$|x\rangle = |x_1\rangle |x_2\rangle, \quad (56)$$

$$|\eta_t(x)\rangle = |\eta_{t_{h_1}}(x)\rangle |\eta_{t_{h_2}}(x)\rangle |\eta_{t_o}(\eta_{t_h})\rangle, \quad (57)$$

$$|t\rangle = |t_{h_1^1}\rangle |t_{h_1^2}\rangle |t_{h_2^1}\rangle |t_{h_2^2}\rangle |t_{o^1}\rangle |t_{o^2}\rangle, \quad (58)$$

where $\eta_{t_{h_j}}(x) = \sigma(t_{h_j^1} x_1 + t_{h_j^2} x_2)$, and similarly for η_o acting on the hidden activations.

2-Layer Representation with Step Activation Function.

We experimented with the architecture shown in Figure 8 in the 2D case, but due to the induced circuit size, we were restricted to a two-layer model with two hidden units and one output unit, using binary weights and no biases. The circuit computes the weighted sum and activates the output qubit (sets it to 1) if this sum does not exceed a hard-coded threshold. This requires three qubits per unit: one weight qubit for each of the two input dimensions and one output qubit. Such a binary 2-layer MLP therefore has six weights in total, corresponding to $2^6 = 64$ possible network configurations.

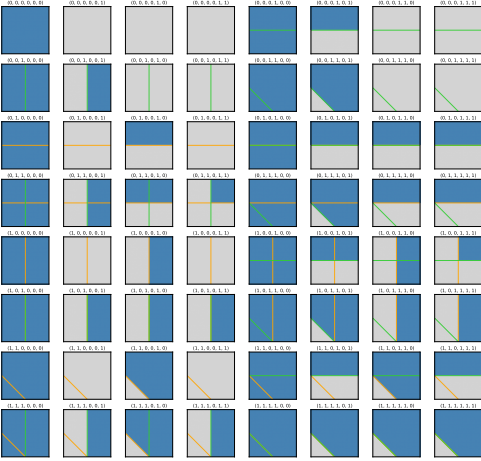
In Figure 9, we visualise the possible activations produced by this binary model across all weight configurations. The dimensional qubit-resolution of x is 6, and the thresholds of the hidden and output units are set to 32 and 1, respectively, in one setting, and to 64 and 1 in the other. Although the network has 64 possible weight assignments, the resulting activations are not very expressive: only seven distinct partitions of the 2D input space appear in the first setting and three in the second. The model would require learnable biases to shift activation boundaries freely, but introducing biases would significantly increase the qubit count. In addition, the hidden activations are no longer needed once they have been used in the output layer. Hence, they unnecessarily grow the qubit count.

In Figure 10, we visualise the image fitting results using the 2-layer MLP as the expressive representation, with thresholds (16, 1) in the hidden and output layers. We observe that the MLP with fixed thresholds still lacks expressivity and does not significantly improve over the standard BVN.

E. Rectangle Expressive Representation

In the following, we provide details on the rectangle activation as an exemplary expressive representation to be used in our generalised BVNs (see Section 4.3).

All 64 Binary Weight Configurations (th_h=[32, 32], th_o=1)



All 64 Binary Weight Configurations (th_h=[64, 64], th_o=1)

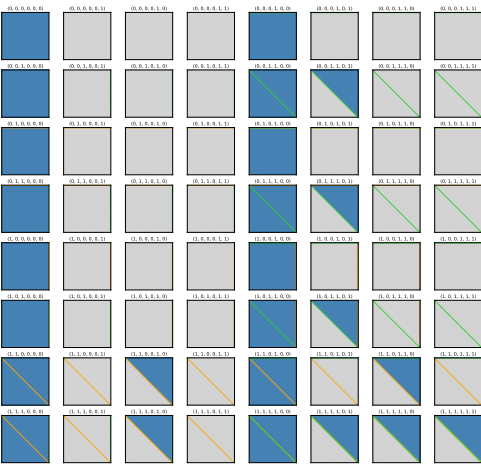


Figure 9. All distinct activation patterns produced by the 2-layer binary MLP for all 64 possible weight configurations. Top: thresholds (32, 1) for hidden and output units. Bottom: thresholds (64, 1). Despite the large number of weight settings, only a small number of unique input-space partitions are realisable, highlighting the limited expressivity of this bias-free architecture. Blue means 1 and grey 0.

To overcome the limited expressivity of bias-free 2-layer MLPs, we introduce the *rectangle activation*. Each input dimension is represented with a binary register, and smaller mask registers define local subdomains. The circuit accumulates contributions from each dimension into a shared sigma register, with superposition allowing all inputs and rectangles to be processed simultaneously.

In Figure 11, we illustrate the circuit implementing the rectangle activation. Since hidden activations are only temporarily needed for computing the output, we use the input register to store them temporarily. Let r_x be the qubit resolution of one dimension x_i of the input x and r_t that of the rectangle positions. Then, the rectangle width is $2^{r_x - r_t}$, and there are 2^{r_t} positions to place the rectangle along the

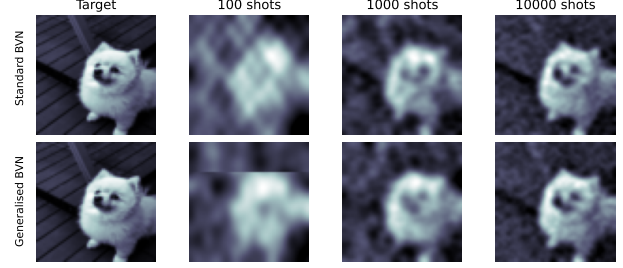


Figure 10. Visualising the fitting results of the standard (top row) and generalised (bottom row) BVNs for different numbers of shots, using the 2-layer MLP as the expressive representation. With few shots, we observe the effect of the MLP, which activates different regions of the input domain. However, the fixed region boundaries limit the model’s ability to capture fine details in the image.

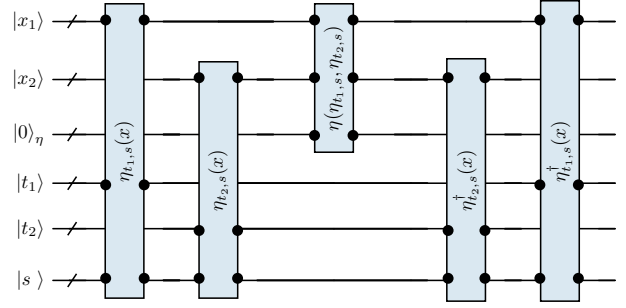


Figure 11. Quantum circuit implementing the rectangle activation. We circularly shift each input dimension x by its corresponding rectangle parameter t , and activate the output qubit if all high-significance qubits of x are 0, which corresponds to activating the rectangle parametrised by t . To reduce artefacts, the superposition register allows for overlapping rectangles by further shifting the input conditionally.

considered dimension i . We circularly shift each input dimension x_i by its corresponding parameter t_i :

$$|x_i\rangle \leftarrow |x_i + t_i \cdot 2^{r_x - r_t} \bmod 2^{r_x}\rangle. \quad (59)$$

The output qubit is activated (set to 1) if all $(r_x - r_t)$ most significant qubits of the input register are 0. After the activation, the shift is undone, leaving the input register back in the x -state. To reduce artefacts, we allow rectangles to overlap by introducing a superposition register $|s\rangle$, which duplicates rectangle parameters. Depending on the value of s , we shift the input further before activation. In our experiments, we use one superposition qubit and shift by half of the rectangle resolution if the superposition bit is 1, yielding

$$|x_i\rangle \leftarrow |x_i + t_i \cdot 2^{r_x - r_t} + s \cdot 2^{r_t/2} \bmod 2^{r_x}\rangle. \quad (60)$$

In Figure 12, we visualise the rectangle activations in 2D. The dimensional qubit-resolution of x is 5, and that of the rectangle positions is 3. With only 6 additional qubits (com-

pared to nine in the MLP case plus ancillaries), the procedure effectively creates localised rectangles in the input space where the activation is turned on, modifying the standard BV basis locally. The result is a highly expressive activation pattern that captures fine-grained variations in the input while remaining resource-efficient.



Figure 12. All distinct activation patterns produced by the rectangle activation for all 64 possible weight configurations. Top: activations for $s = 0$. Bottom: activations for $s = 1$. Each weight configuration generates a distinct local activation, illustrating the expressive power of the rectangle activation without requiring biases or additional qubits. Blue indicates 1 and grey indicates 0.

In Figure 13, we validate the expressivity of the rectangle activation for the resolutions $r_r = 3$ and $r_r = 5$, and ablate the number of shots used in the image fitting experiment. We clearly see how each rectangle locally modifies and corrects the standard BV basis functions toward a better fit of the target. As a consequence, the generalised model converges faster and captures finer details than the standard model. The finer the resolution r_r , the better the results.

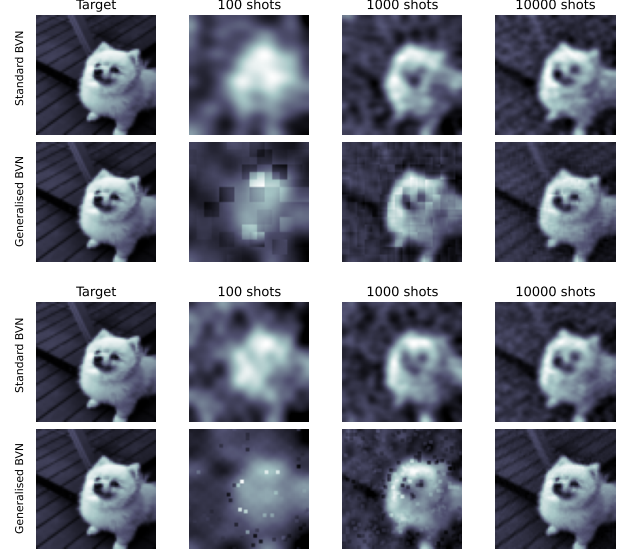


Figure 13. Visualising the fitting results of the standard and generalised BVNs for different numbers of shots and two rectangle resolutions, 3x3 (top) and 5x5 (bottom). With a few shots, we observe the effect of the rectangle activation functions. Each rectangle locally modifies the standard BV basis, allowing the model to better fit the target in that region. With a sufficient number of shots, the rectangles are no longer visible, and the generalised model becomes more accurate than the standard one. The finer rectangle resolution outperforms the coarse.

F. Interference Operators in Comparison

We visualise the activation or basis function induced by the Hadamard, Fourier and Chebyshev interference operators in Figure 14. The Hadamard operator produces step parity functions with amplitude 1, while the Fourier and Chebyshev operators introduce finer variations.

We test these interference operators on regression examples of fitting 1D functions with increasing frequencies. The functions are generated by sampling random vectors of different lengths corresponding to the frequencies and interpolating the vectors. We also include a step function to analyse the Gibbs phenomenon. We compare the Hadamard and Chebyshev interference operators, along with the regularisation factor λ of the coefficient reconstruction. We omit the Fourier transform due to the imaginary part it introduces, which is not needed for the real-valued experiment.

Figure 15 presents the 1D regression results. The Hadamard and Chebyshev operators perform similarly overall, as both can capture the function’s variations. However, the Chebyshev operator, which operates in the real domain, produces smooth outputs, whereas the Hadamard operator yields step-like functions due to rounding effects inherent to the binary domain. Both the standard and generalised models fit the target function reasonably well. Different values of λ smooth the approximation, a property we expect to

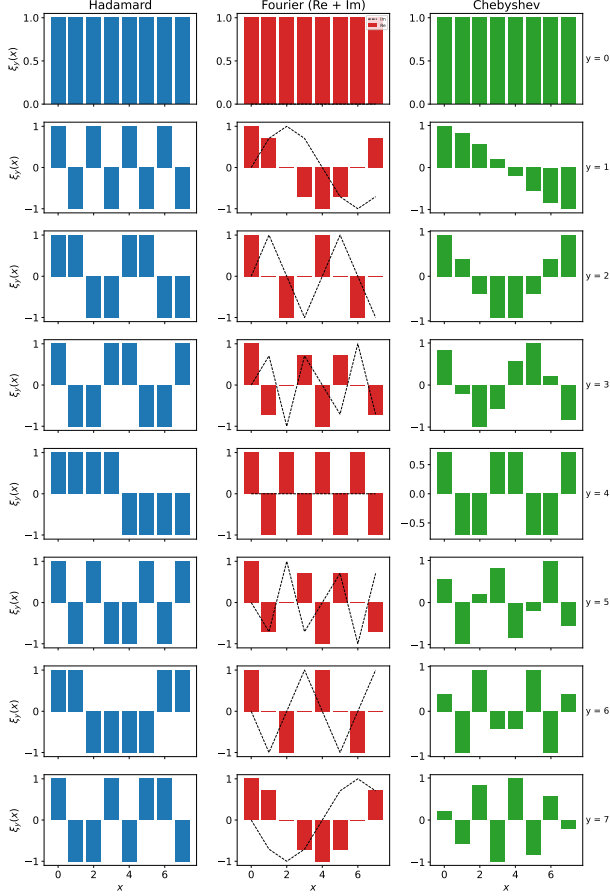


Figure 14. Interference operators in comparison. We visualise the basis functions $\xi_y(\cdot)$ for the Hadamard, the Fourier and the Chebyshev operators. The Hadamard basis functions are parity activations that yield ± 1 depending on whether the weighted sum of the input is odd or even. The Fourier and Chebyshev basis functions, in contrast, allow more variations than ± 1 and smoother activations.

be beneficial in the presence of noise. A closer inspection shows that the generalised model is more accurate and better captures high-frequency variations. For example, with the Chebyshev interference operator applied to the step function (first column, second row), the standard BVN oscillates around the target, whereas the locally corrected basis of the generalised model with rectangles significantly reduces these oscillations and matches the target more closely. See also the image fitting results in Figure 13.

G. Computational Complexity

Our BVNs expect training inputs (binary strings) to be placed in superposition and labelled via an oracle. The gate complexity of this encoding depends on the sparsity of the training set relative to the dimension of the Hilbert space spanned by the input resolution; it is $O(mn)$ for m

being the effective size of the training set and n the number of qubits (details in Section G.1). Further, our rectangle representations with qubit resolution n and n_t for the input and parameter registers use n_t controlled adders and their inverses, along with a Toffoli gate to compute and mark the rectangles in the η -register, yielding a gate complexity of $O(nn_t)$. The Hadamard interference operator can be implemented with $O(n)$ gates on n qubits, whereas Chebyshev (Williams et al., 2023) requires $O(n + n^2)$ gates.

Lastly, for reconstructing the coefficients, we build the matrix $X \in \mathbb{R}^{m \times k}$ by evaluating each of the k sampled basis functions on each of the m training inputs. We then recover the coefficients in closed form as described in Equation (13). Let c denote the cost of evaluating a basis function on a training input, which is $O(n)$ for the Hadamard interference operator and $O(1)$ for the Chebyshev and Fourier interference operators. Then the total cost of reconstructing the coefficients is:

- $O(cmk)$ for constructing X ,
- $O(mk^2)$ for computing the Gram matrix $G = X^\top X$,
- $O(mk)$ for computing $X^\top F$,
- $O(k^3)$ for solving the linear system $(G + \lambda I)^{-1} X^\top F$.

In practice, this reconstruction step was not prohibitively expensive. Its runtime was comparable to a single training epoch in classical models, as it involves a single pass over the training set.

G.1. Dataset-Encoding and Oracle Complexities

Since data encoding is a major bottleneck in QML, we verify that this step is efficient in our BVNs. In our experiments, dataset and oracle steps are combined into an amplitude encoding of a state vector with amplitudes proportional to the label values. As an example, for Iris classification, a 4-dimensional scaled and rounded feature $x = (7, 6, 10, 11)$ is encoded with 4 qubits per dimension as $x = (0111, 0110, 1010, 1011)$, giving when concatenated $x = 0111011010101011$. Each input is thus a 16-qubit computational basis vector $|x\rangle \in \mathbb{C}^{2^{16}}$, and the labelled training set is encoded as a normalised superposition $\sum_x f(x) |x\rangle$ using amplitude encoding. The gate complexity of amplitude encoding depends on the sparsity of the input vector (Schuld & Petruccione, 2018). Sparse-state preparation techniques have a complexity of $O(mn)$ for m nonzero entries (Gleinig & Hoefler, 2021; Ramacciotti et al., 2024), i.e., the effective size of the training set. In practice, $m \ll 2^n$ (150 Iris, 333 Penguins samples for a 2^{16} dimensional space), yielding highly sparse vectors. Even for our dummy-filled datasets in the Iris and Penguins experiments, the algorithm in (Mozafari et al., 2022), based on Decision Diagrams (DDs) is expected applicable, which complexity $O(kn)$,

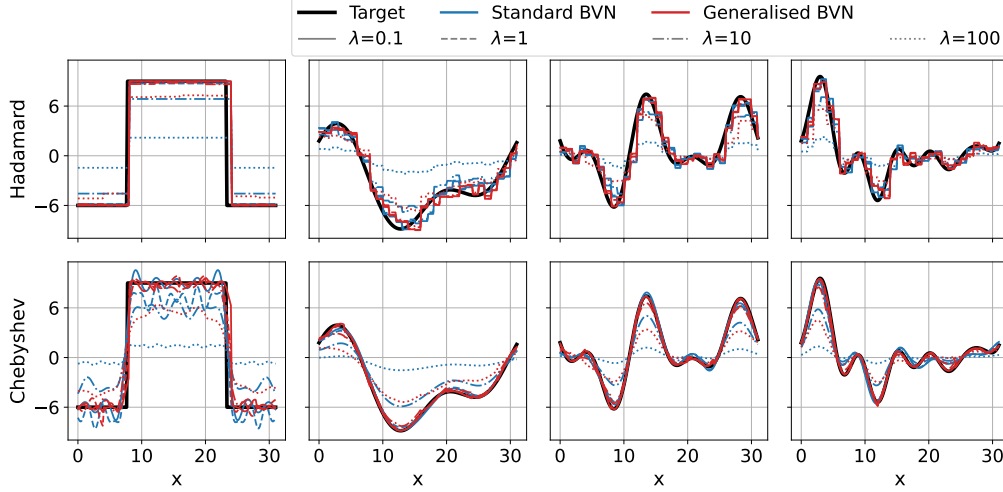


Figure 15. Regression of random 1D functions with different frequencies using the proposed interference-based quantum networks. The top and bottom rows show results with the Hadamard and Chebyshev interference operators, respectively. We compare the standard Bernstein–Vazirani model (blue) with the generalised model (red). Within each model, we contrast the Gram inversion regularisation factors λ . The Chebyshev interference produces smooth functions as it acts in the real domain, while Hadamard, due to rounding, produces step functions as it acts in the binary domain. The generalised method with $\lambda = 0.1$ fits the target function better than that of the standard model (see step function in column 1). Increasing λ regularises the approximation.

where k is the number of DD paths. Our dummy-filled state is dominated by a constant-amplitude component over large regions of the computational basis, with only a few data-dependent deviations, so the effective path count k is expected to scale mainly with m .

The data-loading cost in BVNs compares strictly better than variational QML methods, requiring $O(mn)$ separate state preparation operations per parameter-shift rule evaluation of a single partial derivative, multiplied by the number of shots required for each estimate, multiplied by the number of parameters, and multiplied by the number of training iterations. Concretely, let c denote the number of shots needed to estimate an expectation value, which is required for computing partial derivatives when using the only certified method, the parameter-shift rule. To estimate an expectation value to precision ϵ , it is known that the required number of shots scales as $c = 1/\epsilon^2$. For illustration, a single partial derivative with accuracy $\epsilon = 0.001$ already requires $2c = 2 \cdot 10^6$ circuit evaluations. And we have to multiply this cost by the number of parameters and then by the number of training iterations.

G.2. Sample Complexity Analysis

The number of measurement shots in BVNs is not directly transferable to the approximation accuracy of the target function f , and is also not influenced by the dimension of the input space. Rather, it is determined by the “sparsity” of the target function f in the chosen basis χ .

We can generalise the error bound analysis for the proposed

BVN framework. In fact, since the standard BVN is a special case of the generalised BVN, we can write f uniquely in the standard subspace of the extended basis as

$$f(x) = \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_2^n} \hat{f}(y, 0, 0) \chi_{y00}(x), \quad (61)$$

or, more generally, as the non-unique expansion

$$f(x) = \frac{1}{\sqrt{2^n}} \sum_a \hat{f}(a) \chi_a(x), \quad (62)$$

where $a = (y, z, t)$ indexes the generalised basis functions.

Now let Ω denote the set of sampled basis states of the generalised BVN, and define

$$f_\Omega(x) = \frac{1}{\sqrt{2^n}} \sum_{a \in \Omega} \hat{f}(a) \chi_a(x). \quad (63)$$

Then the approximation error is

$$\epsilon = \mathbb{E}_x [(f(x) - f_\Omega(x))^2] \quad (64)$$

$$= \frac{1}{2^n} \sum_x \left(\frac{1}{\sqrt{2^n}} \sum_{a \notin \Omega} \hat{f}(a) \chi_a(x) \right)^2 \quad (65)$$

$$= \frac{1}{2^n} \sum_x \sum_{a, b \notin \Omega} \hat{f}(a) \hat{f}(b) \chi_a(x) \chi_b(x) \quad (66)$$

$$= \sum_{a, b \notin \Omega} \hat{f}(a) \hat{f}(b) \underbrace{\left(\frac{1}{2^n} \sum_x \chi_a(x) \chi_b(x) \right)}_{=: G_{ab}}, \quad (67)$$

where G is the Gram matrix of the sampled basis functions. Hence,

$$\epsilon = \hat{F}_\Omega^\top G_{\bar{\Omega}} \hat{F}_\Omega, \quad (68)$$

where $\bar{\Omega}$ denotes the complement of Ω . Since $G_{\bar{\Omega}}$ is positive semidefinite,

$$\epsilon \leq \lambda_{\max}(G_{\bar{\Omega}}) \|\hat{f}_\Omega\|_2^2, \quad (69)$$

where $\lambda_{\max}(G_{\bar{\Omega}})$ denotes the largest eigenvalue of the Gram matrix of the generalised basis functions and $\hat{F}_\Omega = (\hat{f}(a))_{a \in \Omega}$ is the vector of residual coefficients.

From the above derivation, we can further derive an explicit bound for the standard case. Indeed, for the standard BVN basis, the Gram matrix satisfies $G = I$ by orthonormality, and therefore $\epsilon = \|\hat{F}_\Omega\|_2^2 = 1 - s$, where $s = \sum_{a \in \Omega} |\hat{f}(a)|^2$ denotes the accumulated spectral mass. Thus, in the standard BVN, the approximation error is determined entirely by the unsampled Fourier spectrum.

For the generalised BVN, the induced basis is generally over-complete and therefore non-orthogonal. Consequently, the approximation error satisfies $\epsilon \leq \lambda_{\max}(G_{\bar{\Omega}}) \|\hat{F}_\Omega\|_2^2$. This bound exposes a natural trade-off introduced by the expressive representation. The more correlated the basis functions are, the higher the penalty factor $\lambda_{\max}(G_{\bar{\Omega}})$. At the same time, the richer representation η_t enables the target function to admit more compact representations in the extended basis, so that the norm of the residual coefficients $\|\hat{F}_\Omega\|_2^2$ may decrease substantially.

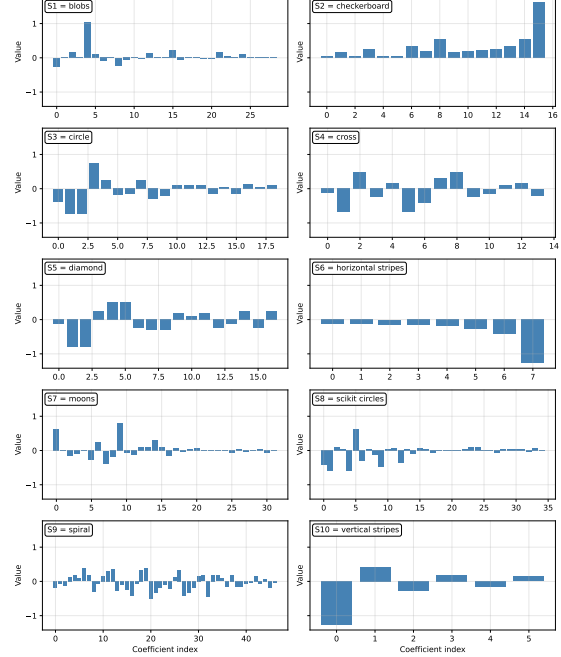
Our empirical results of shots-vs-accuracy in Figure 7 show consistently lower approximation errors for the generalised BVNs, suggesting that this reduction in $\|\hat{F}_\Omega\|_2^2$ dominates the additional Gram-matrix factor.

H. Inspecting the Sampled Coefficients

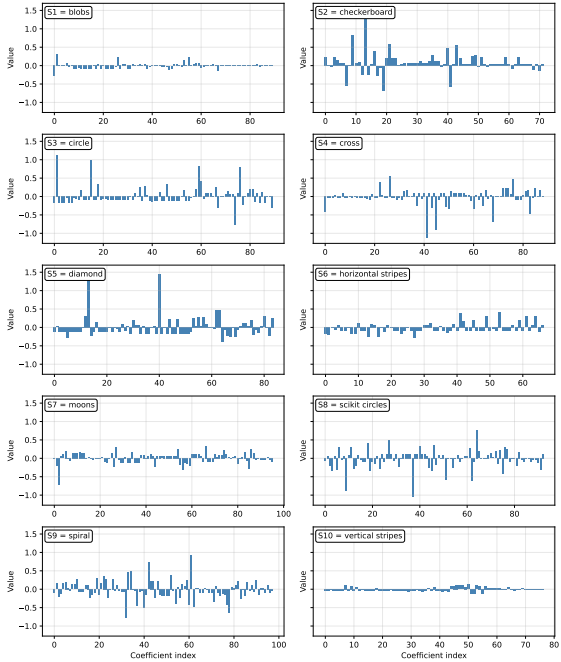
We inspect the histograms of the coefficients sampled by the standard and generalised BVNs for the classification task on the 2D synthetic datasets Figures 16a and 16b, as well as on the real-world Iris and Penguins datasets Figure 17.

From the histograms of the standard BVN Figures 16a and 17, one can infer how easily the objective function can be approximated on a given dataset. Sparse histograms indicate that the target function is sparse in the BV basis, i.e., relatively easy to approximate with a finite number of shots. More challenging datasets, such as *spiral*, exhibit more dispersed histograms.

The generalised model Figures 16b and 17 does not produce sparse histograms. This is expected, as it extends the standard BV basis and generates multiple copies of the dominant BV functions, each with a local modification.



(a) Standard BVN



(b) Generalised BVN

Figure 16. Histograms of sampled coefficients for the 2D synthetic datasets using (a) the standard BVN and (b) the generalised BVN.

I. Comparing Filled and Unfilled Training Data

As mentioned in the main text (Section 5.2), the small number of training samples (< 500) relative to the large

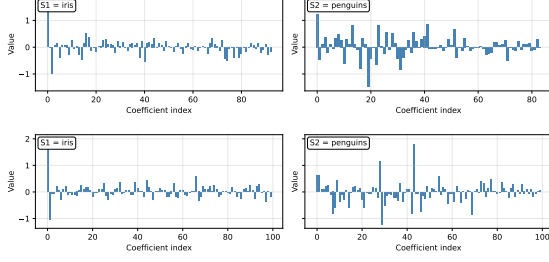


Figure 17. Histograms of the sampled coefficients obtained with the standard (top row) and generalised BV (bottom row) methods on the Iris and Penguins datasets. The coefficients are not concentrated and exhibit dispersed magnitudes.

Hilbert-space dimension of 10^{16} (4 qubits per input dimension) leads to sampling leakage (Wakeham & Schuld, 2024), which led us to fill the dataset. In this section, we complement the filled and unfilled comparison in Figure 5 by reporting the performance for additional values of λ .

In Figure 18, we experimentally assign a dummy fill-label to domain inputs not present in the training set. This small change substantially improves the generalised model, while the standard model degrades. The poor performance of the standard model on unfilled data is now due not to leakage but to the dummy label overwhelming the true training samples, causing the model to assign it uniformly as a constant approximation of the target function. The generalised model, using the same number of shots, gains enough expressivity to capture the geometry of the true training data.

J. Future Work and Technical Extensions

While the Generalised Bernstein–Vazirani model demonstrates strong capabilities on synthetic data, scaling to complex, real-world distributions (e.g., Iris, Penguins) presents challenges related to spectral leakage and shot complexity.

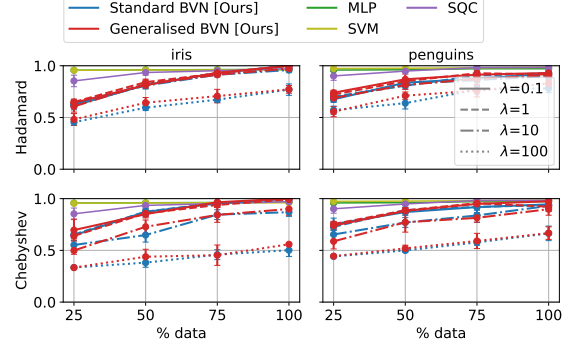
The primary source of error in current experiments is the misalignment between the discrete, hard-edged quantum basis and the continuous geometry of natural data. We propose two enhancements to the expressive representation.

J.1. Adaptive Basis Alignment

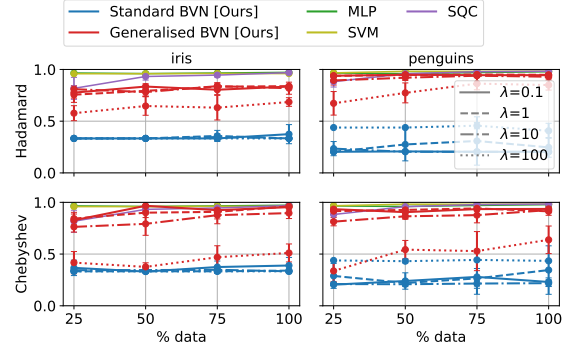
Standard interference bases are static, determined solely by fixed parameters. To align the basis with the dataset’s principal manifolds, we propose a hybrid variational protocol. We introduce a parametrised unitary $V(\theta)$ applied to the parameter register $|t\rangle$ prior to interference.

The objective is to maximise the sparsity of the measurement distribution $P(y, z, t|\theta)$. We define the cost function $\mathcal{L}(\theta)$ as the Shannon entropy:

$$\mathcal{L}(\theta) = - \sum_{y, z, t} P(y, z, t|\theta) \log P(y, z, t|\theta). \quad (70)$$



(a) Unfilled training data



(b) Filled training data

Figure 18. Classification results for unfilled (a) and filled (b) training data on the Iris and Penguins datasets for different λ . Filling the data strongly affects the performance of the BVNs: Unfilled data lead to poor generalisation, whereas filled data, while degrading the standard model, substantially improve the generalised model, allowing $> 90\%$ accuracy with only 25% training data, comparable to the baselines.

A classical pre-training loop minimises $\mathcal{L}(\theta)$, rotating the basis manifold to maximise constructive interference for the specific target dataset.

J.2. Apodised Activation Functions

The current rectangle activation $\chi_{rect}(x)$ acts as a boxcar function, exhibiting high-frequency leakage (Gibbs phenomenon). We propose replacing the binary thresholding with **apodised activations**.

By replacing the multi-controlled Toffoli in the activation step with a controlled rotation $CR_y(\phi)$ on the ancilla, we can implement “soft” trapezoidal windows:

$$U_{activ}|x\rangle|a\rangle = |x\rangle \left(\cos \frac{\phi(x)}{2} \mathbb{I} - \sin \frac{\phi(x)}{2} iY \right) |a\rangle, \quad (71)$$

where $\phi(x)$ varies smoothly across the boundary margin δ . This acts as a low-pass filter in feature space, accelerating spectral decay to $O(1/\omega^2)$ and concentrating signal energy.