Supplementary Information: Photonic Quantum-Enhanced Knowledge Distillation

Kuan-Cheng Chen[1,2,8], Shang Yu[2,3], Chen-Yu Liu[4], Samuel Yen-Chi Chen[5], Huan-Hsin Tseng[5], Yen Jui Chang[6,7], Wei-Hao Huang[8], Felix Burt[1,2], Esperanza Cuenca Gomez[9], Zohim Chandani[9], William Clements[10], Ian Walmsley[2,3,11], Kin K. Leung[1]

Kuan-Cheng Chen and Shang Yu contributed equally to this work.

kuan-cheng.chen17@imperial.ac.uk shang.yu@imperial.ac.uk
d10245003@g.ntu.edu.tw ycchen1989@ieee.org htseng@bnl.gov
aceest@cycu.edu.tw w.huang@j-ij.com f.burt23@imperial.ac.uk
ecuencagomez@nvidia.com zchandani@nvidia.com wclements@orcacomputing.com
ian.walmsley@physics.ox.ac.uk kin.leung@imperial.ac.uk

[1] Department of Electrical and Electronic Engineering, Imperial College London, South Kensington, London SW7 2AZ, England, United Kingdom
[2] Imperial Centre for Quantum Engineering, Science and Technology, Imperial College London, South Kensington, London SW7 2AZ, England, United Kingdom
[3] Blackett Laboratory, Department of Physics, Imperial College London, South Kensington, London SW7 2AZ, England, United Kingdom
[4] Graduate Institute of Applied Physics, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 106319, Taiwan
[5] Computational Science Initiative, Brookhaven National Laboratory, Bldg. 725, Room 2-189, P.O. Box 5000, Upton, NY 11973-5000, USA
[6] Quantum Information Center, Chung Yuan Christian University, No. 200, Zhongbei Rd., Zhongli Dist., Taoyuan City 320314, Taiwan
[7] Master Program in Intelligent Computing and Big Data, Chung Yuan Christian University, No. 200, Zhongbei Rd., Zhongli Dist., Taoyuan City 320314, Taiwan
[8] JIJ Inc., Rutherford Appleton Laboratory, Harwell Campus, Didcot OX11 0QX, United Kingdom; and 3-3-6 Shibaura, Minato-ku, Tokyo 108-0023, Japan
[9] NVIDIA Corporation, Santa Clara, CA, USA
[10] ORCA Computing, London, United Kingdom
[11] Department of Physics, University of Oxford, Parks Road, Oxford OX1 3PU, England, United Kingdom

Supplementary Contents

S1 Fundamentals

S1.1 Knowledge distillation

Knowledge distillation [33, 46] trains a compact student model to approximate the predictive behaviour of a stronger teacher by using the teacher's *soft* class probabilities as supervision. Let the teacher produce logits $t(x) \in \mathbb{R}^C$ and the student produce logits $s(x; w, \theta) \in \mathbb{R}^C$ for an input x, where C is the number of classes, w denotes the trainable student parameters, and θ denotes the photonic parameters that condition the student through the feature $z(\theta)$ described in Supplementary Note S2. For temperature $\tau > 0$, define softened distributions

$$p_T^{(\tau)}(x) = \mathrm{softmax}\left(\frac{t(x)}{\tau}\right), \qquad p_S^{(\tau)}(x; w, \theta) = \mathrm{softmax}\left(\frac{s(x; w, \theta)}{\tau}\right). \tag{S1}$$

Larger τ increases the entropy of the distributions, revealing "dark knowledge" in the teacher's inter-class similarities, which can be particularly informative when the student hypothesis class is strongly constrained by compression.

We optimise the student using a standard distillation objective that interpolates between hard-label supervision and teacher matching:

$$\mathscr{L}_{\mathrm{KD}}(w, \theta) = \mathbb{E}_{(x,y)}\Big[\lambda\ \mathrm{CE}\big(y,\ p_S^{(1)}(x; w, \theta)\big) + (1 - \lambda)\,\tau^2\ \mathrm{KL}\big(p_T^{(\tau)}(x)\ \|\ p_S^{(\tau)}(x; w, \theta)\big)\Big], \tag{S2}$$

where $\lambda \in [0, 1]$ controls the mixture and $\mathrm{CE}(y, p)$ denotes cross-entropy between the one-hot label y and predicted probability vector p. The τ^2 factor preserves gradient magnitudes as τ varies, because the softmax gradients scale approximately as $1/\tau$; without this factor, the distillation signal would diminish at higher temperatures.

The Kullback–Leibler term admits an equivalent and practically useful interpretation. Using $\mathrm{KL}(p\|q) = \sum_i p_i \log \frac{p_i}{q_i}$, we can write

$$\mathrm{KL}\big(p_T^{(\tau)} \| p_S^{(\tau)}\big) = -\sum_{c=1}^{C} p_{T,c}^{(\tau)}(x)\ \log p_{S,c}^{(\tau)}(x; w, \theta) + \sum_{c=1}^{C} p_{T,c}^{(\tau)}(x)\ \log p_{T,c}^{(\tau)}(x), \tag{S3}$$

that is,

$$\mathrm{KL}\big(p_T^{(\tau)} \| p_S^{(\tau)}\big) = \mathrm{CE}\big(p_T^{(\tau)},\ p_S^{(\tau)}\big) - \mathrm{H}\big(p_T^{(\tau)}\big), \tag{S4}$$

where $\mathrm{H}(p_T^{(\tau)})$ is the Shannon entropy of the teacher distribution and is independent of the student parameters. Consequently, minimising the KL term is equivalent to minimising cross-entropy against the teacher soft targets[33, 47]. This encourages the student to reproduce the teacher's *relative class structure* (including non-argmax probabilities), which provides a denser training signal than one-hot labels and empirically improves optimisation and generalisation in compressed regimes.

Finally, in the PQKD setting the distillation signal plays an additional structural role: because the student parameters are restricted to a low-dimensional manifold induced by the photonic-conditioned convolution parameterisation, the teacher soft targets guide the optimisation toward solutions that best approximate the teacher

within this constrained model class. In this sense, distillation acts as the mechanism that selects, via (w, θ), a high-performing point on the compression–accuracy frontier reported in the Sec. 2.2 and Sec. 2.3.

S1.2 Model compression and structured parameterisation

Modern deep networks are frequently over-parameterised relative to the complexity of the target task, yet the resulting parameter budgets can be prohibitive in settings where *training-time* updates, deployment constraints, or communication overhead dominate[48]. In particular, for edge inference[49, 50], federated learning[51], continual learning[52], and hardware-in-the-loop optimisation[53], the cost of maintaining and updating large weight tensors can exceed the cost of forward computation. Compression methods therefore aim to reduce resource requirements while preserving accuracy, and can be broadly categorised according to the resource being optimised.

We distinguish three complementary notions of compression:

- **Trainable-parameter compression:** reducing the number of parameters that must be learned or updated to reach a given performance level. This directly impacts optimisation complexity, storage of trainable state (including optimiser moments), and the communication cost of gradient/parameter synchronisation[54, 55].
- **Inference compute reduction:** reducing the floating-point operations (FLOPs), latency, or energy consumed during inference. This is commonly targeted by pruning, low-rank factorisation, or efficient architectural blocks[56, 57].
- **Bandwidth/memory reduction:** reducing the cost of storing, transmitting, or streaming model weights and activations, which can be a dominant bottleneck on modern accelerators due to limited memory bandwidth[58, 59].

These notions are related but not equivalent: for example, a method can sharply reduce trainable parameters while leaving inference FLOPs largely unchanged, or vice versa. In this work, we focus on *trainable-parameter compression* because PQKD is designed as a structured mechanism that restricts the effective degrees of freedom of convolutional layers while allowing the resulting compressed hypothesis class to be tuned by a compact photonic control signal.

Our approach follows the principle of *structured parameterisation*: rather than zeroing or quantising weights post hoc, we explicitly constrain the weight tensors to lie in a low-dimensional family that is efficient to store and update. Convolutional layers are particularly suitable for this strategy because their parameters factor naturally into spatial and channel dimensions. By replacing dense kernels with a dictionary of spatial basis filters and a low-dimensional mixing mechanism (Supplementary Note S2), PQKD reduces the number of trainable degrees of freedom in the convolutional stack. Knowledge distillation then supplies a dense supervisory signal that mitigates the accuracy loss typically associated with strong constraints, enabling us to trace a controlled compression–accuracy frontier.

S1.3 Continuous-variable photonic quantum computing

Continuous-variable (CV) photonic quantum computing provides a hardware platform in which information is encoded in bosonic modes of the electromagnetic field and processed through optical transformations[18, 19]. In integrated implementations, the dominant programmable components are reconfigurable interferometric networks composed of beam splitters and phase shifters, optionally augmented by Gaussian operations such as squeezing (when available)[13, 60]. From a learning-systems perspective, a defining characteristic is that readout is naturally *sampling-based*: photon-number-resolving (PNR) detection[61] or threshold detection[62, 63] produces stochastic outcomes whose empirical statistics converge to an underlying measurement distribution as the shot budget S increases.

Photonic quantum knowledge distillation (PQKD) is designed to interface directly with this CV photonic sampling model. Rather than requiring analytic gradients through the photonic circuit—which can be challenging under realistic noise, imperfect calibration, or restricted access to device internals—PQKD treats the photonic processor as a controllable stochastic feature generator. Circuit parameters θ modulate the output distribution, and a fixed classical feature map converts finite-shot samples into a compact conditioning vector that controls a structured parameterisation of the student network. This separation respects near-term photonic hardware constraints: the photonic device is used in the regime it naturally supports (shot-limited sampling from a programmable optical circuit), while optimisation of the coupled classical–photonic system proceeds with sampling-robust, gradient-free updates for θ and standard gradient-based updates for the classical student parameters.

A CV photonic circuit can be described as a sequence of elementary gate operations acting on one or more bosonic modes. Each mode is associated with annihilation and creation operators $(\hat{a}, \hat{a}^\dagger)$, number operator $\hat{n} = \hat{a}^\dagger\hat{a}$, and quadratures $(\hat{x}, \hat{p})$[19]. In practice, the gate set implemented in integrated photonics is dominated by *Gaussian* operations—those generated by Hamiltonians at most quadratic in $(\hat{a}, \hat{a}^\dagger)$—including displacements, rotations, squeezing, and multi-mode linear interferometers built from beam splitters and phase shifters[60, 64]. Non-Gaussian gates (generated by higher-than-quadratic Hamiltonians), such as Kerr interactions, are more challenging experimentally but are conceptually important because Gaussian operations alone are not universal for quantum computation over CV systems[20, 65]. In the experiments considered here, the photonic module is parameterised through a programmable interferometer (a product of beam splitters and rotations) and is accessed through sampling-based measurement; therefore, this gate-level description provides a direct bridge between physical programmability and the algorithmic interface[62, 66].

Gaussian single- and two-mode gates.

The elementary Gaussian gates used to construct most CV photonic processors include[18]:

- **Displacement.** The displacement gate shifts the phase-space mean of a mode and can be written as

$$\hat{D}(\alpha) = \exp\big(\alpha\hat{a}^\dagger - \alpha^*\hat{a}\big), \qquad \alpha \in \mathbb{C}, \tag{S5}$$

or equivalently parameterised as $\alpha = ae^{i\theta}$ with amplitude $a \geq 0$ and phase $\theta \in [0, 2\pi)$.

- **Squeezing.** The squeezing gate changes the variances of conjugate quadratures and generates nonclassical states:

$$\hat{S}(s) = \exp\Big(\tfrac{1}{2}(s^*\hat{a}^2 - s\,\hat{a}^{\dagger 2})\Big), \qquad s \in \mathbb{C}, \tag{S6}$$

often written as $s = re^{i\phi}$ with squeezing strength $r \geq 0$ and phase $\phi \in [0, 2\pi)$.

- **Rotation (phase shift).** The rotation gate corresponds to an optical phase shift:

$$\hat{R}(\theta) = \exp(-i\theta\hat{n}), \qquad \theta \in [0, 2\pi), \tag{S7}$$

and implements a rotation in phase space, $(\hat{x}, \hat{p}) \mapsto (\hat{x}\cos\theta + \hat{p}\sin\theta,\ -\hat{x}\sin\theta + \hat{p}\cos\theta)$.

- **Beam splitter.** A beam splitter couples two modes through interference:

$$\hat{BS}(\theta, \phi) = \exp\Big(\theta\big(e^{i\phi}\hat{a}_1\hat{a}_2^\dagger - e^{-i\phi}\hat{a}_1^\dagger\hat{a}_2\big)\Big), \qquad \theta, \phi \in [0, 2\pi), \tag{S8}$$

and is the fundamental two-mode primitive used to build general multi-mode linear interferometers.

A representative non-Gaussian gate.

A canonical example of a non-Gaussian CV gate is the Kerr interaction[20],

$$\hat{\Phi}(\kappa) = \exp(i\kappa\hat{n}^2), \qquad \kappa \in [0, 2\pi), \tag{S9}$$

which is generated by a quartic Hamiltonian and can induce highly nonclassical dynamics. While such nonlinearities are generally more demanding to realise in scalable integrated platforms, they are relevant conceptually because adding a suitable non-Gaussian element to Gaussian operations enables universal CV quantum computing. In the present work, we do not rely on explicit Kerr nonlinearities; instead, we exploit sampling-based measurement and classical post-processing as the operational interface to the photonic module.

Interferometers as products of beam splitters and rotations.

A key workhorse in CV photonic hardware is the N-mode linear interferometer, which implements a unitary transformation on the mode operators. In the lossless case, this transformation can be compiled into a sequence of beam splitters and single-mode rotations (phase shifts). Thus, a programmable interferometer can be represented abstractly as

$$\hat{U}(\boldsymbol{\vartheta}) = \prod_{\ell=1}^{L} \hat{BS}(\theta_\ell, \phi_\ell) \prod_{m=1}^{M} \hat{R}(\varphi_m), \tag{S10}$$

where $\boldsymbol{\vartheta}$ collects all tunable angles. This is precisely the type of parameterisation that underlies many integrated meshes [64, 67] and loop-based time-bin architectures[68];

Table S1 Common CV gate operations in photonic circuits. Gaussian gates are generated by Hamiltonians at most quadratic in $(\hat{a}, \hat{a}^\dagger)$; Kerr is a representative non-Gaussian gate. In many programmable photonic processors, $\hat{U}$ is implemented as a product of beam splitters and rotations forming an N-port interferometer.

Name	Gate	Unitary	Parameters	Gaussian	Modes
Displacement	$\hat{D}(\alpha)$	$\exp(\alpha\hat{a}^\dagger - \alpha^*\hat{a})$	$\alpha \in \mathbb{C}$ (or a, θ)	✓	1
Squeezing	$\hat{S}(s)$	$\exp\big(\frac{1}{2}(s^*\hat{a}^2 - s\hat{a}^{\dagger 2})\big)$	$s \in \mathbb{C}$ (or r, ϕ)	✓	1
Rotation	$\hat{R}(\theta)$	$\exp(-i\theta\hat{n})$	$\theta \in [0, 2\pi)$	✓	1
Beam splitter	$\hat{BS}(\theta, \phi)$	$\exp(\theta(e^{i\phi}\hat{a}_1\hat{a}_2^\dagger - e^{-i\phi}\hat{a}_1^\dagger\hat{a}_2))$	$\theta, \phi \in [0, 2\pi)$	✓	2
Kerr	$\hat{\Phi}(\kappa)$	$\exp(i\kappa\hat{n}^2)$	$\kappa \in [0, 2\pi)$	×	1

in our implementation, this structure manifests as a constraint on the number of beam-splitter parameters per "tiling" (Supplementary Note S5).

This gate-level view clarifies how PQKD interfaces with CV photonics. The photonic module is parameterised by tunable interferometric angles (implemented physically through phase shifters and couplers) and is queried through repeated measurement. Rather than requiring explicit analytic gradients through the gate sequence, PQKD uses finite-shot samples to construct a stable feature vector $z(\theta)$, which conditions a structured convolutional parameterisation in the student network. This design aligns the algorithm with the dominant near-term CV photonic capabilities: programmable Gaussian interferometers and sampling-based detection.

S2 Definition of Photonic Quantum Knowledge Distillation

S2.1 Photonic quantum feature map

The role of the photonic module in PQKD is to provide a compact, hardware-aligned *conditioning signal* for the compressed student network[24]. Operationally, the photonic processor is accessed through repeated measurement: for a fixed input preparation and a programmable optical circuit, we observe finite-shot samples from an output distribution whose statistics depend on a low-dimensional parameter vector θ[69]. PQKD converts these samples into a deterministic, fixed-length feature vector that can be used by standard deep-learning components without requiring an analytic differentiable model of the photonic device.

Formally, let ρ_{in} denote a fixed N-mode input state and let $U(\theta)$ be a parameterised CV unitary representing the programmable photonic transformation. The output state is

$$\rho_\theta = U(\theta)\,\rho_{\text{in}}\,U(\theta)^\dagger. \tag{S11}$$

A measurement described by a positive operator-valued measure (POVM) $\{E_\omega\}_{\omega\in\Omega}$ induces a probability distribution over outcomes $\omega \in \Omega$,

$$p_\theta(\omega) = \mathrm{Tr}\big(E_\omega \rho_\theta\big), \tag{S12}$$

which captures all device-level effects relevant to the algorithmic interface. In practice, the distribution is not accessed directly; instead, we obtain S independent shots $\omega_1, \ldots, \omega_S \sim p_\theta$ and construct an empirical distribution (histogram) $\widehat{p}_\theta$. The photonic quantum feature map is then defined as

$$z(\theta) = \Phi(\widehat{p}_\theta) \in \mathbb{R}^d, \tag{S13}$$

where Φ is a fixed classical post-processing rule that (i) aggregates shot-limited outcomes into a stable representation, (ii) produces a constant-dimensional vector compatible with neural-network layers, and (iii) is inexpensive relative to the downstream classical training.

This design makes explicit that the photonic contribution enters PQKD through *distributional statistics* rather than single-shot values: $z(\theta)$ summarises the circuit-dependent measurement behaviour and serves as a low-dimensional control signal for structured parameter generation in the student network (Supplementary Note S2). The shot budget S sets the statistical fidelity of $\widehat{p}_\theta$ and therefore governs the variance of $z(\theta)$; this dependence motivates the noise and stability analyses reported in Sec. 2.4 and the use of variance-reduction strategies such as feature smoothing (Supplementary Note S6).

In the main experiments, we use $N = 16$ modes and construct a $d = 512$ dimensional feature vector by concatenating two 256-bin marginal histograms derived from 8-bit partitions of thresholded detection outcomes (Supplementary Note S5). This choice balances representational richness with robustness under finite shots, and yields a fixed interface that remains unchanged across compression configurations (compressing one, two, or all convolutional layers).

S2.2 Photonic-conditioned dictionary convolution

PQKD compresses convolutional layers by replacing a fully trainable kernel tensor with a *structured, low-dimensional parameterisation* whose effective degrees of freedom are controlled by the photonic feature vector $z(\theta)$. This section provides a complete mathematical specification of the parameterisation, its implementation-friendly tensor shapes, and its trainable-parameter accounting.

Standard convolution.

Consider an input activation map $x \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ and an output activation map $h \in \mathbb{R}^{C_{\text{out}} \times H' \times W'}$. A standard 2D convolution layer with kernel size $k \times k$ and padding p_{pad} is parameterised by a weight tensor

$$W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k \times k}, \tag{S14}$$

and optional bias $b \in \mathbb{R}^{C_{\text{out}}}$. The output is

$$h_o = \sum_{i=1}^{C_{\text{in}}} W_{o,i} * x_i + b_o, \qquad o = 1, \ldots, C_{\text{out}}. \tag{S15}$$

where $*$ denotes spatial convolution and $W_{o,i} \in \mathbb{R}^{k\times k}$ is the kernel slice coupling input channel i to output channel o.

Dictionary factorisation of kernels.

PQKD replaces the dense kernel W by a dictionary (basis) of R spatial filters and a set of channel mixing coefficients. Specifically, introduce trainable basis filters

$$B \in \mathbb{R}^{R\times k\times k}, \tag{S16}$$

and mixing coefficients

$$M \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times R}, \tag{S17}$$

such that each channel-to-channel kernel is expressed as a linear combination of the shared basis:

$$W_{o,i,:,:} = \sum_{r=1}^{R} M_{o,i,r}\, B_{r,:,:}. \tag{S18}$$

Equation (S18) is a *structured* parameterisation: spatial degrees of freedom are concentrated in the basis B, while channel coupling is handled by M. If $R \ll C_{\text{out}}C_{\text{in}}$, this can drastically reduce the number of trainable parameters *provided that* M itself is not fully trainable.

Photonic-conditioned mixing (core PQKD mechanism).

The key idea in PQKD is to avoid training the full mixing tensor M. Instead, M is generated deterministically from the photonic feature $z(\theta) \in \mathbb{R}^d$ via a fixed linear map:

$$\text{vec}(M) = A\, z(\theta), \qquad A \in \mathbb{R}^{(C_{\text{out}}C_{\text{in}}R)\times d}. \tag{S19}$$

Here $\text{vec}(\cdot)$ denotes vectorisation that stacks all entries of M into a column vector. The matrix A is *fixed* (sampled once using a prescribed random seed and then held constant) and is therefore excluded from trainable parameter counts. Under this construction, the channel mixing coefficients M are not free parameters; they vary only through the d-dimensional photonic control signal $z(\theta)$.

For implementation, one typically reshapes $Az(\theta)$ back into the tensor form

$$M(z) = \text{reshape}\big(Az,\ [C_{\text{out}}, C_{\text{in}}, R]\big). \tag{S20}$$

Combining Eq. (S18) with Eq. (S20) yields an explicit mapping from $z(\theta)$ and the basis B to the full kernel:

$$W(z,B)_{o,i,:,:} = \sum_{r=1}^{R} \Big(\text{reshape}(Az,\ [C_{\text{out}}, C_{\text{in}}, R])\Big)_{o,i,r} B_{r,:,:}. \tag{S21}$$

In other words, *all* channel mixing across the layer is confined to the d-dimensional subspace $\{Az : z \in \mathbb{R}^d\}$.

Equivalent matrix formulation.

Let $W_{(k)} \in \mathbb{R}^{(C_{\text{out}}C_{\text{in}})\times k^2}$ denote the kernel tensor flattened over spatial dimensions, and let $B_{(k)} \in \mathbb{R}^{R\times k^2}$ denote the basis filters flattened similarly. Define $M_{(c)} \in \mathbb{R}^{(C_{\text{out}}C_{\text{in}})\times R}$ as the mixing matrix obtained by flattening the first two indices of M. Then Eq. (S18) is equivalent to

$$W_{(k)} = M_{(c)}\, B_{(k)}. \tag{S22}$$

Equation (S19) specifies that $M_{(c)}$ lies in a d-dimensional linear subspace parameterised by $z(\theta)$. This makes explicit how PQKD imposes a *low-dimensional manifold* constraint on the kernel space.

Forward computation.

Given $z(\theta)$ and basis filters B, the layer computes:

1. Generate $M \leftarrow \text{reshape}(Az(\theta), [C_{\text{out}}, C_{\text{in}}, R])$.
2. Form the effective kernel using Eq. (S18).
3. Apply convolution as in Eq. (S15).

In practice, step (2) can be implemented efficiently via tensor contractions. One convenient expression is

$$W(z, B) = \text{einsum}\big(\texttt{oir,rxy} \to \texttt{oixy},\ M(z),\ B\big), \tag{S23}$$

followed by a standard convolution call $\text{conv2d}(x, W, b, \text{padding} = p_{\text{pad}})$.

Trainable-parameter accounting and compression ratio.

For a single compressed convolution layer, the trainable parameters are:

$$\#\text{params}_{\text{train}} = \underbrace{Rk^2}_{\text{basis } B} + \underbrace{C_{\text{out}}}_{\text{bias}}, \tag{S24}$$

plus the photonic parameter budget $\dim(\theta)$ if θ is included in the overall student budget. In contrast, a dense convolution has

$$\#\text{params}_{\text{dense}} = C_{\text{out}}C_{\text{in}}k^2 + C_{\text{out}}. \tag{S25}$$

Ignoring bias terms for intuition, the per-layer trainable-parameter compression factor is approximately

$$\text{CR}_{\text{layer}} \approx \frac{C_{\text{out}}C_{\text{in}}k^2}{Rk^2} = \frac{C_{\text{out}}C_{\text{in}}}{R}, \qquad (R \ll C_{\text{out}}C_{\text{in}}). \tag{S26}$$

When multiple layers share the same photonic feature $z(\theta)$ (as in our implementation), θ becomes a *shared* overhead that does not scale with layer width, which is advantageous when compressing multiple convolutional layers simultaneously.

Interpretation and relation to hypernetworks.

Equation (S19) can be viewed as a constrained hypernetwork that generates convolutional mixing coefficients from a compact input $z(\theta)$. The constraint that the generator is linear and fixed (through A) intentionally shifts learning capacity away from the generator and into (i) the trainable basis S and (ii) the photonic parameters θ that control $z(\theta)$ through the sampling-based feature map (Supplementary Note S2.1). This design choice reduces trainable degrees of freedom, improves reproducibility, and makes the photonic contribution explicit and measurable through the dependence on θ and the shot budget S.

Extension to multiple compressed layers.

For a network with multiple compressed convolutional layers $\ell \in \mathscr{L}$, each layer has its own $(C_{\text{in}}^{(\ell)}, C_{\text{out}}^{(\ell)}, k_\ell, R_\ell)$, basis $S^{(\ell)}$, and fixed projection $A^{(\ell)}$, while sharing the same $z(\theta)$:

$$\text{vec}(M^{(\ell)}) = A^{(\ell)} z(\theta), \qquad W^{(\ell)}_{o,i,:,:} = \sum_{r=1}^{R_\ell} M^{(\ell)}_{o,i,r}\, B^{(\ell)}_{r,:,:}. \tag{S27}$$

This yields a coherent mechanism to study "compress conv1", "compress conv1+conv2", and "compress all convs" settings under a unified control signal and parameter-counting convention.

S2.3 Compression configurations

To quantify how PQKD scales with compression strength and network depth, we consider three structured compression regimes that progressively replace larger fractions of the convolutional stack by photonic-conditioned dictionary convolutions (Supplementary Note S2.2). Throughout, the teacher network uses standard dense convolutions, while the student network matches the teacher macro-architecture (same number of blocks, pooling schedule, and head) but substitutes selected convolution layers by the PQKD parameterisation. This design isolates the effect of structured parameterisation and the photonic conditioning signal from confounding architectural changes.

Regime I: compress `conv1` only.

In this regime, only the first convolution layer is replaced by the photonic-conditioned dictionary convolution:

$$\text{vec}(M^{(1)}) = A^{(1)} z(\theta), \qquad W^{(1)}_{o,i,:,:} = \sum_{r=1}^{R_1} M^{(1)}_{o,i,r}\, S^{(1)}_{r,:,:}. \tag{S28}$$

All subsequent convolutions (e.g., `conv2` and `conv3`) remain dense and fully trainable. This setting represents a conservative compression strategy that primarily reduces parameters near the input while leaving downstream representational capacity largely intact. It provides a baseline to test whether PQKD can compress early feature extraction without destabilising training.

Regime II: compress `conv1+conv2`.

Here, both the first and second convolution layers are replaced by PQKD layers with (potentially distinct) basis ranks R_1 and R_2:

$$\mathrm{vec}(M^{(\ell)}) = A^{(\ell)} z(\theta), \qquad W^{(\ell)}_{o,i,:,:} = \sum_{r=1}^{R_\ell} M^{(\ell)}_{o,i,r}\, S^{(\ell)}_{r,:,:}, \qquad \ell \in \{1,2\}. \tag{S29}$$

The third convolution remains dense. This regime substantially increases the fraction of trainable convolution parameters controlled by $z(\theta)$ while still retaining one fully trainable convolution block to absorb residual modelling error. Empirically, this intermediate regime often yields a favourable trade-off between compression and accuracy because it compresses a large portion of the convolutional stack while maintaining a flexible downstream adaptation layer.

Regime III: compress all convolution layers (`conv1--conv3`).

In the most aggressive setting, *all* convolution layers are replaced by photonic-conditioned dictionary convolutions:

$$\mathrm{vec}(M^{(\ell)}) = A^{(\ell)} z(\theta), \qquad W^{(\ell)}_{o,i,:,:} = \sum_{r=1}^{R_\ell} M^{(\ell)}_{o,i,r}\, S^{(\ell)}_{r,:,:}, \qquad \ell \in \{1,2,3\}. \tag{S30}$$

This regime tests large-scale structured compression, in which the convolutional feature extractor is entirely restricted to the PQKD manifold defined by $\{A^{(\ell)} z(\theta)\}$ and the trainable bases $\{S^{(\ell)}\}$. Since the same $z(\theta)$ conditions every compressed layer, the photonic parameters θ act as a *shared global control* over the entire convolutional stack, while each layer retains local spatial expressivity through its basis $S^{(\ell)}$ and rank R_ℓ.

Rank selection and interpretability.

Each compressed layer ℓ has an associated basis rank R_ℓ, which controls the number of spatial basis filters in that layer. Intuitively, larger R_ℓ increases the spatial expressivity of the reconstructed kernels (Eq. (S18)) at the cost of more trainable parameters ($R_\ell k_\ell^2$). In contrast, the channel mixing degrees of freedom remain confined to the d-dimensional subspace induced by $A^{(\ell)} z(\theta)$ (Eq. (S19)). Our sweep studies therefore treat $\{R_\ell\}$ and $\dim(\theta)$ as primary compression knobs.

Parameter accounting across regimes.

For a student model with compressed layers $\mathcal{L}$, the total trainable parameters are

$$\#\mathrm{params}_{\mathrm{student}} = \sum_{\ell \in \mathcal{L}} \left(R_\ell k_\ell^2 + C^{(\ell)}_{\mathrm{out}}\right) + \sum_{\ell \notin \mathcal{L}} \left(C^{(\ell)}_{\mathrm{out}} C^{(\ell)}_{\mathrm{in}} k_\ell^2 + C^{(\ell)}_{\mathrm{out}}\right) + \#\mathrm{params}_{\mathrm{head}} + \dim(\theta), \tag{S31}$$

where $\#\mathrm{params}_{\mathrm{head}}$ denotes the parameters of the non-convolutional head (e.g., global-average-pooling classifier). We report both an *overall* compression ratio (teacher

parameters divided by student parameters, including $\dim(\theta)$) and a *conv-only* compression ratio restricted to the convolutional layers, to disentangle compression achieved in the feature extractor from contributions of the classifier head.

Practical implementation.

In all regimes, each compressed layer uses its own fixed projection matrix $A^{(\ell)}$ (sampled once and held constant) and its own trainable basis $S^{(\ell)}$, while sharing the same photonic feature vector $z(\theta)$ across layers. This choice keeps the student trainable parameter count low and makes the effect of θ and the shot budget S directly comparable across compression regimes.

S3 Theory—expressivity, approximation, and stability

S3.1 Trainable-parameter accounting

To report compression ratios consistently, we distinguish *trainable* parameters (those updated by learning) from *fixed* parameters (constants sampled once and held throughout training). This distinction is essential for PQKD because the channel-mixing generator uses a fixed projection matrix A (Eq. (S19)), which can be large in memory but does not contribute to the number of learnable degrees of freedom.

Dense convolution (baseline).

A standard 2D convolution with kernel size $k \times k$, C_{in} input channels and C_{out} output channels has trainable parameters

$$\#\text{params}_{\text{dense}} = C_{\text{out}} C_{\text{in}} k^2 + C_{\text{out}}, \tag{S32}$$

where the first term is the weight tensor $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k \times k}$ and the second term is the bias vector $b \in \mathbb{R}^{C_{\text{out}}}$.

PQKD compressed convolution.

In PQKD, a compressed convolution layer uses trainable spatial basis filters $B \in \mathbb{R}^{R \times k \times k}$ and a bias $b \in \mathbb{R}^{C_{\text{out}}}$, while the channel mixing coefficients are generated deterministically as $\text{vec}(M) = Az(\theta)$ with fixed A and photonic feature $z(\theta)$. Therefore, the trainable parameters *per compressed layer* are

$$\#\text{params}_{\text{PQKD layer}} = \underbrace{Rk^2}_{\text{basis } B} + \underbrace{C_{\text{out}}}_{\text{bias}}, \tag{S33}$$

and this count is independent of C_{in} because channel mixing is not learned directly.

Global photonic parameter budget.

The photonic parameters $\theta \in \mathbb{R}^{\dim(\theta)}$ are shared across all compressed layers through the feature vector $z(\theta)$. When reporting *overall* student trainable parameters, we

include this shared budget once:

$$\#\text{params}_{\text{student}} = \sum_{\ell \in \mathcal{L}} \left(R_\ell k_\ell^2 + C_{\text{out}}^{(\ell)}\right) + \sum_{\ell \notin \mathcal{L}} \left(C_{\text{out}}^{(\ell)} C_{\text{in}}^{(\ell)} k_\ell^2 + C_{\text{out}}^{(\ell)}\right) + \#\text{params}_{\text{head}} + \dim(\theta), \tag{S34}$$

where $\mathcal{L}$ denotes the set of compressed convolution layers and $\#\text{params}_{\text{head}}$ denotes any non-convolutional parameters (e.g., the GAP classifier head).

Per-layer compression factor.

A useful diagnostic is the per-layer trainable-parameter compression factor, obtained by comparing Eq. (S32) and Eq. (S33). Ignoring the shared $\dim(\theta)$ overhead (which is amortised across layers), we define

$$\text{CR}_{\text{conv}} = \frac{\#\text{params}_{\text{dense}}}{\#\text{params}_{\text{PQKD layer}}} \approx \frac{C_{\text{out}} C_{\text{in}} k^2 + C_{\text{out}}}{R k^2 + C_{\text{out}}} = \frac{C_{\text{out}}(C_{\text{in}} k^2 + 1)}{R k^2 + C_{\text{out}}}. \tag{S35}$$

When $C_{\text{out}} C_{\text{in}} k^2 \gg C_{\text{out}}$ and $R k^2 \gg C_{\text{out}}$ (a common regime for moderate-to-large channel counts), this simplifies to the intuitive approximation

$$\text{CR}_{\text{conv}} \approx \frac{C_{\text{out}} C_{\text{in}}}{R}. \tag{S36}$$

This highlights the role of R as the primary knob controlling trainable degrees of freedom in the compressed convolution.

Reporting conventions used in this work.

We report two complementary ratios:

- **Conv-only ratio:** computed by restricting Eq. (S34) to convolution layers only. This isolates compression achieved in the feature extractor.
- **Overall ratio (including θ):** computed using Eq. (S34), including $\dim(\theta)$ once, to reflect the full trainable budget required by PQKD.

Fixed projection matrices $\{A^{(\ell)}\}$ are excluded from trainable counts throughout because they are not updated during training; however, we note that they can contribute to memory footprint and can be implemented implicitly (e.g., via structured random features) if memory becomes a constraint.

Mixing subspace induced by a fixed generator.

For a given layer with mixing tensor $M \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times R}$, define $m = \text{vec}(M) \in \mathbb{R}^p$ where

$$p := C_{\text{out}} C_{\text{in}} R. \tag{S37}$$

PQKD generates m as

$$m = Az, \qquad A \in \mathbb{R}^{p \times d}, \quad z \in \mathbb{R}^d. \tag{S38}$$

Therefore, all attainable mixing vectors lie in the column space of A:

$$m \in \mathrm{Range}(A) := \{Az : z \in \mathbb{R}^d\}. \tag{S39}$$

If A has full column rank (a typical case when $d \leq p$ and A is drawn from a continuous distribution), then $\dim(\mathrm{Range}(A)) = d$.

Best achievable approximation to a target mixing.

Let $M^\star$ denote an arbitrary target mixing tensor (e.g., the mixing that would reproduce a desired dense kernel given a fixed basis). Let $m^\star = \mathrm{vec}(M^\star) \in \mathbb{R}^p$. The best approximation within the PQKD subspace is the solution of the least-squares problem

$$z^\star = \arg\min_{z \in \mathbb{R}^d} \|m^\star - Az\|_2^2. \tag{S40}$$

When A has full column rank, the minimiser is unique and given by

$$z^\star = (A^\top A)^{-1} A^\top m^\star, \tag{S41}$$

and the corresponding attainable mixing is the orthogonal projection of $m^\star$ onto $\mathrm{Range}(A)$:

$$Az^\star = \Pi_{\mathrm{Range}(A)}\, m^\star, \qquad \Pi_{\mathrm{Range}(A)} := A(A^\top A)^{-1} A^\top. \tag{S42}$$

The residual $\|m^\star - Az^\star\|_2$ quantifies the *subspace approximation error* introduced by restricting M to $\mathrm{Range}(A)$.

Implications for expressivity and the role of d and R.

Two independent design choices control the representational capacity of a compressed convolution:

- **Mixing dimension d:** Increasing d enlarges $\mathrm{Range}(A)$ and reduces the best-case subspace approximation error in Eq. (S42), enabling richer channel coupling patterns across $(C_{\mathrm{out}}, C_{\mathrm{in}})$ for a fixed rank R.
- **Basis rank R:** Increasing R enlarges the dictionary $S \in \mathbb{R}^{R \times k \times k}$ and thus increases *spatial* expressivity in the kernel reconstruction $W = MS$ (Eq. (S22)). Even with perfect mixing, a small R limits the diversity of spatial filters that can be expressed.

Consequently, d and R play complementary roles: d controls how flexibly PQKD can represent channel-to-channel mixing coefficients, whereas R controls how flexibly it can represent spatial kernel structure. Our sweep studies therefore treat $(d, \dim(\theta))$ and $\{R_\ell\}$ as primary axes when constructing compression–accuracy frontiers (Sec. 2.3).

Connection to the photonic quantum feature map.

In PQKD, the vector z is not a free optimisation variable but is obtained as $z(\theta) = \Phi(\widehat{p}_\theta)$ from shot-limited samples (Supplementary Note S2.1). Thus, Eq. (S42) characterises an *idealised best case*: it describes the closest mixing achievable if z could be chosen arbitrarily in $\mathbb{R}^d$. In practice, the attainable set is further constrained by

the photonic map $\theta \mapsto z(\theta)$ and finite-shot variability; these effects are investigated empirically via shot and noise sweeps (Sec. 2.4).

S3.2 Approximation within the mixing subspace

Fix the basis B and consider any target mixing tensor $M^\star$. Under Eq. (S38), the best achievable mixing corresponds to the least-squares projection of $\mathrm{vec}(M^\star)$ onto the column space of A:

$$z^\star = \arg\min_{z\in\mathbb{R}^d} \|\mathrm{vec}(M^\star) - Az\|_2^2, \quad \Rightarrow \quad Az^\star = \Pi_{\mathrm{Range}(A)}\,\mathrm{vec}(M^\star), \tag{S43}$$

where $\Pi_{\mathrm{Range}(A)}$ denotes orthogonal projection. This characterises how d and the choice of A control the representational subspace for channel mixing, while R controls the spatial expressivity through the basis filters.

S3.3 Finite-shot noise propagation

The photonic feature vector $z(\theta)$ is computed from a finite number of measurement shots. Consequently, the PQKD student does not receive the exact distributional feature $z(\theta) = \Phi(p_\theta)$, but rather an empirical estimate $\widehat{z}(\theta) = \Phi(\widehat{p}_\theta)$ constructed from S i.i.d. samples. This section formalises how finite-shot fluctuations propagate through the feature map and into the reconstructed convolution kernels, providing a principled basis for the noise studies reported in Sec. 2.3.

Concentration of the empirical measurement distribution.

Let $\widehat{p}_\theta$ be the empirical histogram over outcomes $\omega \in \Omega$ formed from S independent samples $\omega_1, \ldots, \omega_S \sim p_\theta$. For any fixed bin ω, the random variable $\widehat{p}_\theta(\omega)$ is the sample mean of Bernoulli trials with mean $p_\theta(\omega)$. Hoeffding's inequality yields, for any $\epsilon > 0$,

$$\mathbb{P}(|\widehat{p}_\theta(\omega) - p_\theta(\omega)| \geq \epsilon) \leq 2\exp\left(-2S\epsilon^2\right). \tag{S44}$$

This makes explicit the $1/\sqrt{S}$ scaling of shot noise at the level of individual histogram entries.

From histogram noise to feature noise.

The photonic feature is obtained via a deterministic map Φ applied to the empirical histogram: $\widehat{z}(\theta) = \Phi(\widehat{p}_\theta)$ and $z(\theta) = \Phi(p_\theta)$. Suppose Φ is Lipschitz with respect to the ℓ_1 norm on histograms, i.e.,

$$\|\Phi(q) - \Phi(q')\|_2 \leq L_\Phi \|q - q'\|_1 \qquad \text{for all histograms } q, q'. \tag{S45}$$

This assumption is satisfied by histogram-based feature constructions composed of linear operations, normalisation away from degenerate cases, and bounded affine rescaling; it holds for our marginal-histogram mapping under standard numerical

stabilisers (Supplementary Note S5). Applying Eq. (S45) gives

$$\|\widehat{z}(\theta) - z(\theta)\|_2 \leq L_\Phi \|\widehat{p}_\theta - p_\theta\|_1. \tag{S46}$$

To interpret $\|\widehat{p}_\theta - p_\theta\|_1$, let K denote the effective number of histogram bins used by Φ (e.g., $K = 512$ for two concatenated 256-bin marginals). Standard multinomial concentration bounds imply that

$$\mathbb{E}[\|\widehat{p}_\theta - p_\theta\|_1] = \mathcal{O}\left(\sqrt{\frac{K}{S}}\right), \tag{S47}$$

and therefore

$$\mathbb{E}[\|\widehat{z}(\theta) - z(\theta)\|_2] = \mathcal{O}\left(L_\Phi \sqrt{\frac{K}{S}}\right). \tag{S48}$$

Equation (S48) provides a direct link between shot budget S, feature dimension K, and the magnitude of stochastic feature perturbations seen by the student network.

Propagation to mixing coefficients and convolution kernels.

In PQKD, the mixing tensor is generated linearly from the feature:

$$\mathrm{vec}(M) = Az, \qquad \mathrm{vec}(\widehat{M}) = A\widehat{z}, \tag{S49}$$

so the induced perturbation obeys

$$\|\mathrm{vec}(\widehat{M}) - \mathrm{vec}(M)\|_2 \leq \|A\|_2 \, \|\widehat{z} - z\|_2. \tag{S50}$$

Next, the reconstructed kernel satisfies $W = MB$ in the flattened formulation (Eq. (S22)). Using submultiplicativity of operator norms, one obtains a Lipschitz bound for the kernel perturbation:

$$\|W(B, \widehat{z}) - W(B, z)\|_F = \|\widehat{M}B - MB\|_F \leq \|\widehat{M} - M\|_F \, \|B\|_2 \leq \|A\|_2 \, \|B\|_F \, \|\widehat{z} - z\|_2. \tag{S51}$$

where the final inequality uses $\|B\|_2 \leq \|S\|_F$ and the relationship between Frobenius and Euclidean norms under vectorisation.

Implications for training stability and noise studies.

Equations (S48)–(S51) show that the effective kernel noise induced by finite shots scales approximately as $\mathcal{O}(\sqrt{K/S})$, amplified by the projection magnitude $\|A\|_2$ and the basis energy $\|B\|_F$. This motivates two practical considerations that we evaluate empirically: (i) increasing S reduces stochasticity in the photonic conditioning signal, and (ii) variance-reduction strategies such as feature smoothing (e.g., EMA in Supplementary Note S6) can reduce the high-frequency fluctuations of $\widehat{z}(\theta)$, thereby stabilising both the induced kernels and the downstream student optimisation. These predictions are directly tested in the shot and noise impact experiments in the Sec. 2.4.

S4 Classical models and baselines

This Supplementary Note specifies the classical teacher and student architectures used to evaluate PQKD, together with the baseline models required for fair attribution. Our design principle is to control for confounding factors: the teacher and PQKD student share the same high-level topology (number of convolutional blocks, pooling schedule, and classifier head), and we vary *only* the parameterisation of selected convolutional kernels and the source of the conditioning signal. This ensures that observed performance differences can be attributed to structured compression and photonic conditioning, rather than to unrelated architectural changes.

S4.1 Teacher CNN (global-average-pooling head)

The teacher network is a conventional convolutional classifier with three convolutional blocks and a global-average-pooling (GAP) head[70], chosen for (i) strong performance on MNIST/CIFAR-10 without excessive depth, (ii) stable training under standard optimisers, and (iii) compatibility with structured compression of the convolutional stack. Let C_{in} denote the input channels and let (c_1, c_2, c_3) denote the channel widths.

Architecture.

The teacher consists of:

- **conv1:** $C_{\text{in}} \to c_1$ with 5×5 kernels, padding 2, followed by ReLU and 2×2 max-pooling.
- **conv2:** $c_1 \to c_2$ with 3×3 kernels, padding 1, followed by ReLU and 2×2 max-pooling.
- **conv3:** $c_2 \to c_3$ with 3×3 kernels, padding 1, followed by ReLU and dropout with $p = 0.25$.
- **GAP:** adaptive average pooling to spatial size 1×1.
- **Classifier:** a linear layer mapping $c_3 \to C$ with $C = 10$ classes.

Teacher width sweep.

To obtain clear compression ratios while maintaining a strong teacher reference, we sweep teacher widths over representative settings such as

$$(c_1, c_2, c_3) \in \{(32, 64, 128),\ (48, 96, 128),\ (64, 128, 128)\}, \tag{S52}$$

and report teacher accuracy and parameter counts for each setting. In all cases, the teacher is trained to convergence (or a fixed epoch budget) using standard cross-entropy on hard labels.

Teacher parameter count (convolutional stack).

For completeness, the number of trainable parameters in the three convolutional layers is

$$\#\text{params}_{\text{teacher,conv}} = \sum_{\ell=1}^{3} \left(C_{\text{out}}^{(\ell)} C_{\text{in}}^{(\ell)} k_\ell^2 + C_{\text{out}}^{(\ell)} \right), \tag{S53}$$

with $(k_1, k_2, k_3) = (5, 3, 3)$ and $(C_{\text{out}}^{(1)}, C_{\text{out}}^{(2)}, C_{\text{out}}^{(3)}) = (c_1, c_2, c_3)$.

S4.2 Student architectures

The PQKD student matches the teacher macro-architecture (same block structure and GAP head) but replaces selected convolutional layers by photonic-conditioned dictionary convolutions (Supplementary Note S2.2). Concretely, for each compressed convolution layer $\ell \in \mathscr{L}$, the dense kernel is replaced by a trainable spatial basis $S^{(\ell)} \in \mathbb{R}^{R_\ell \times k_\ell \times k_\ell}$ and a photonic-generated mixing tensor $M^{(\ell)}$:

$$\text{vec}(M^{(\ell)}) = A^{(\ell)} z(\theta), \qquad W^{(\ell)}_{o,i,:,:} = \sum_{r=1}^{R_\ell} M^{(\ell)}_{o,i,r} \, S^{(\ell)}_{r,:,:}, \qquad \ell \in \mathscr{L}. \tag{S54}$$

Layers not in $\mathscr{L}$ remain standard dense convolutions with fully trainable kernels. We evaluate three compression configurations (Supplementary Note S2.3): compress `conv1` only; compress `conv1+conv2`; and compress all convolution layers.

S4.3 Baselines and controls

Because PQKD combines two ingredients—structured parameterisation and a photonic conditioning signal—we include baseline models that isolate each factor. The goal is to distinguish: (i) gains attributable to distillation alone, (ii) gains attributable to dictionary structure alone, and (iii) gains attributable specifically to photonic-conditioned parameter generation.

1. **KD-only student (no compression).** A student with reduced width (or identical architecture) trained using the distillation objective in Eq. (S2), without replacing any convolution kernels. This baseline quantifies the benefit of knowledge distillation absent structured compression.
2. **Dictionary-only student (no photonic conditioning).** Replace the target convolution layers by the same dictionary decomposition $W_{o,i} = \sum_r M_{o,i,r} S_r$, but treat M as a *trainable* parameter tensor rather than generating it from $z(\theta)$. This isolates the effect of the dictionary factorisation while removing the photonic control constraint.
3. **Random-feature conditioning.** Replace the photonic feature $z(\theta)$ by i.i.d. Gaussian features $\tilde{z} \sim \mathcal{N}(0, I_d)$ (optionally re-sampled per epoch or fixed per run), while keeping the same fixed projection A and dictionary basis parameterisation. This baseline tests whether PQKD improvements could be explained by generic stochastic conditioning rather than photonic structure.
4. **Fixed photonic feature (no θ learning).** Use the photonic sampling and feature map to compute $z(\theta)$ but keep θ fixed throughout training (no SPSA or other updates). This baseline isolates the value of *learning* θ as opposed to using a static photonic feature.

When reporting results, we keep training budgets (epochs, batch sizes, optimiser settings) identical across PQKD and baselines unless explicitly stated, and we use the same teacher checkpoint for all student variants in a given experimental run.

S4.4 Parameter counting conventions

We report compression ratios under two complementary conventions to avoid ambiguity:

- **Overall trainable-parameter ratio:**

$$\mathrm{CR}_{\mathrm{overall}} = \frac{\#\mathrm{params}_{\mathrm{teacher}}}{\#\mathrm{params}_{\mathrm{student}}} \quad \text{with } \#\mathrm{params}_{\mathrm{student}} \text{ including } \dim(\theta) \text{ once.} \tag{S55}$$

- **Conv-only ratio:**

$$\mathrm{CR}_{\mathrm{conv}} = \frac{\#\mathrm{params}_{\mathrm{teacher,conv}}}{\#\mathrm{params}_{\mathrm{student,conv}}}, \tag{S56}$$

  computed by restricting the parameter count to convolutional layers only. This isolates compression achieved in the feature extractor independent of the classifier head.

Fixed projection matrices $A^{(\ell)}$ are excluded from all trainable-parameter counts because they are not optimised; however, they may contribute to memory footprint and can be implemented in structured form if required. Photonic parameters θ are included in the student trainable budget by default (unless explicitly stated otherwise) to reflect the full learnable budget of the PQKD mechanism.

S5 Quantum model details

S5.1 Circuit family and theta-length constraint

Our PQKD implementation interfaces with a programmable CV photonic sampler whose native parameterisation is tied to a specific interferometric circuit family. In particular, the backend adopts a *tiled* interferometer structure consistent with loop- or time-bin architectures, where each "tiling" (or layer) implements a fixed-pattern mode mixing described by a sequence of beam-splitter operations interleaved with phase shifts. Within this circuit family, the number of independent mixing angles per tiling scales linearly with the number of modes N, rather than quadratically as in a fully general $U(N)$ mesh. This architectural choice reflects realistic constraints of compact photonic implementations (e.g., single-loop interferometers), and it directly determines the admissible length of the continuous parameter vector θ accepted by the sampler.

Per-tiling parameter count.

For an N-mode instance, a single tiling is parameterised by

$$L_{\mathrm{tile}} = N - 1 \tag{S57}$$

real-valued angles associated with beam-splitter reflectivities (and/or equivalent mixing parameters) in the fixed connectivity pattern of the backend. For the main

experiments with $N = 16$ modes, this implies

$$L_{\text{tile}} = N - 1 = 15. \tag{S58}$$

We denote by $\theta^{(t)} \in \mathbb{R}^{L_{\text{tile}}}$ the parameter vector controlling tiling t.

Tiling interpretation for extended parameter vectors.

To support systematic sweeps over the effective photonic parameter budget $\dim(\theta)$, we allow user-specified vectors whose length may exceed L_{tile}. When $\dim(\theta) > L_{\text{tile}}$, the backend interprets the circuit as a concatenation of multiple identical connectivity tiles, each with its own set of $(N-1)$ parameters. Specifically, we define the number of tiles as

$$n_{\text{tiling}} = \left\lceil \frac{\dim(\theta)}{N-1} \right\rceil, \tag{S59}$$

so that the circuit consumes a total of

$$L_{\text{eff}} = (N-1)\, n_{\text{tiling}} \tag{S60}$$

angles. Conceptually, this corresponds to composing the interferometer as a product

$$U(\theta_{\text{fit}}) = U^{(n_{\text{tiling}})}(\theta^{(n_{\text{tiling}})}) \cdots U^{(2)}(\theta^{(2)}) U^{(1)}(\theta^{(1)}), \tag{S61}$$

where each $U^{(t)}$ is a single-tiling interferometric transformation parameterised by $\theta^{(t)} \in \mathbb{R}^{N-1}$.

Deterministic adjustment for sampler compatibility.

Because the sampler expects exactly L_{eff} parameters, we map the user-specified $\theta \in \mathbb{R}^{\dim(\theta)}$ to a backend-compatible vector

$$\theta \mapsto \theta_{\text{fit}} \in \mathbb{R}^{(N-1)\, n_{\text{tiling}}}, \tag{S62}$$

using a deterministic rule that preserves reproducibility:

$$\theta_{\text{fit}} = \begin{cases} [\theta;\, \mathbf{0}] & \text{if } \dim(\theta) < L_{\text{eff}} \quad \text{(zero-padding)}, \\ \theta_{1:L_{\text{eff}}} & \text{if } \dim(\theta) > L_{\text{eff}} \quad \text{(truncation)}, \\ \theta & \text{if } \dim(\theta) = L_{\text{eff}}. \end{cases} \tag{S63}$$

Here $[\theta; \mathbf{0}]$ denotes concatenation with zeros. This adjustment guarantees that every call to the photonic sampler receives a valid parameter vector, while allowing us to treat $\dim(\theta)$ as a controlled hyperparameter in compression sweeps.

Implementation details and practical considerations.

For numerical stability and physical interpretability of the underlying interferometric angles, we additionally apply a fixed bounding operation to θ_{fit} when required by the backend (e.g., wrapping angles to a principal interval):

$$\theta_{\text{fit}} \leftarrow \text{wrap}(\theta_{\text{fit}}; [-\pi, \pi]) \quad \text{or} \quad \theta_{\text{fit}} \leftarrow \text{clip}(\theta_{\text{fit}}; [-\theta_{\max}, \theta_{\max}]), \tag{S64}$$

where the specific convention is held constant across all runs. Importantly, the tiling expansion in Eq. (S59) increases the effective circuit depth (number of repeated mixing stages) without changing the mode count N, and therefore provides a principled way to scale the photonic control capacity while maintaining a fixed feature dimension d and a fixed sampling interface.

Overall, this theta-length constraint is not merely a software artefact: it reflects a physically motivated parameterisation of a structured interferometric circuit family. Our deterministic mapping in Eq. (S63) ensures strict sampler compatibility, reproducible hyperparameter sweeps over $\dim(\theta)$, and a clear interpretation of additional photonic parameters as additional interferometric tiles.

S5.2 Measurement and sample format

PQKD accesses the photonic processor through repeated measurements. Each circuit execution (shot) produces a multi-mode detection outcome whose statistics depend on the circuit parameters θ and the chosen input state preparation. To make the downstream feature construction well-defined and hardware-aligned, we adopt a standard *thresholded* measurement representation that converts raw detection events into a fixed-length binary vector over modes.

Measurement model.

Let $\rho_\theta = U(\theta)\rho_{\text{in}}U(\theta)^\dagger$ be the output state of the N-mode photonic circuit. A single shot applies a measurement described by a POVM $\{E_\omega\}_{\omega\in\Omega}$ and returns a random outcome $\omega \sim p_\theta(\omega) = \text{Tr}(E_\omega \rho_\theta)$. In photon-counting settings[63], ω can be represented as a photon-number pattern

$$n = (n_1, \ldots, n_N) \in \mathbb{Z}_{\geq 0}^N, \tag{S65}$$

where n_j is the number of detected photons in mode j. In threshold-detection settings, the detector reports only whether each mode is "clicked" (occupied) or "not clicked" (empty)[62], which can be viewed as a coarse-graining of photon-number measurements.

Thresholding to a binary occupancy vector.

For consistency across backends and to reduce sensitivity to rare high-count events, we convert each raw pattern n into a binary occupancy vector

$$b = (b_1, \ldots, b_N) \in \{0, 1\}^N, \qquad b_j = \mathbb{I}[\, n_j > 0\,], \tag{S66}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Thus, each shot yields a length-N bitstring b encoding per-mode occupancy. We denote the resulting random variable by $B \sim P_\theta$ over $\{0,1\}^N$, where P_θ is the induced distribution after thresholding.

Finite-shot sampling and empirical distribution.

For a single feature evaluation, we collect S independent samples

$$b^{(1)}, \ldots, b^{(S)} \overset{\text{i.i.d.}}{\sim} P_\theta, \tag{S67}$$

and compute empirical statistics (e.g., histograms of selected marginals) to form $\widehat{p}_\theta$ and the corresponding feature vector $z(\theta) = \Phi(\widehat{p}_\theta)$ (Supplementary Note S2.1). In all experiments, S is treated as an explicit hyperparameter controlling the trade-off between feature variance and evaluation cost, and the impact of finite-shot noise is studied in Supplementary Note S3.3 and Supplementary Note 2.4.

Implementation.

In code, the sampler returns either an array of per-shot samples of shape (S, N) with entries in $\{0,1\}$ (after thresholding), or an equivalent representation (e.g., integer-encoded bitstrings) that can be deterministically converted to the binary format in Eq. (S66). We standardise all outputs to the binary matrix representation to ensure that feature construction and parameter generation are backend-agnostic.

S5.3 Feature construction

This subsection specifies the exact photonic feature map Φ used in the main experiments. The guiding design goal is to transform shot-limited binary detection patterns into a *fixed-length*, *order-consistent*, and *variance-controlled* representation that (i) is simple to implement, (ii) remains stable under finite-shot sampling, and (iii) preserves meaningful distributional information beyond per-mode means.

Bitstring partitioning.

Each shot produces a thresholded occupancy vector $b \in \{0,1\}^N$ with $N = 16$ modes (Supplementary Note S5.2). We partition the bitstring into two contiguous 8-bit halves,

$$b = \left[b^{(1)}; b^{(2)}\right], \qquad b^{(1)} \in \{0,1\}^8, \quad b^{(2)} \in \{0,1\}^8. \tag{S68}$$

This produces two low-dimensional views (marginals) of the 16-bit outcome. Importantly, this construction avoids the exponential (2^{16}) histogram that would be too sparse under realistic shot budgets, while still capturing structured multi-bit correlations within each half.

Index mapping.

For each half $b^{(m)} = (b_1^{(m)}, \ldots, b_8^{(m)})$, we define a deterministic integer encoding

$$\mathrm{idx}\left(b^{(m)}\right) = \sum_{j=1}^{8} b_j^{(m)}\, 2^{8-j} \in \{0, 1, \ldots, 255\}, \qquad m \in \{1, 2\}, \tag{S69}$$

where the bit ordering convention (most-significant to least-significant) is fixed throughout all runs. Any consistent convention is valid as long as it is held constant across training and evaluation.

Empirical marginal histograms.

Given S i.i.d. samples $\{b_s\}_{s=1}^{S}$, we compute the two empirical marginal histograms $h^{(1)}, h^{(2)} \in \mathbb{R}^{256}$:

$$h^{(m)}[k] = \frac{1}{S}\sum_{s=1}^{S} \mathbf{1}\Big\{\mathrm{idx}\Big(b_s^{(m)}\Big) = k\Big\}, \qquad k \in \{0,\dots,255\}, \quad m \in \{1,2\}, \tag{S70}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. By construction, each histogram is a valid empirical probability mass function:

$$h^{(m)}[k] \geq 0, \qquad \sum_{k=0}^{255} h^{(m)}[k] = 1. \tag{S71}$$

Concatenation and standardisation.

We form a 512-dimensional feature by concatenating the two marginals:

$$\tilde{z}(\theta) = \big[h^{(1)};\, h^{(2)}\big] \in \mathbb{R}^{512}. \tag{S72}$$

To improve numerical conditioning and to make the scale of the conditioning signal comparable across runs, we apply an affine standardisation:

$$z(\theta) = \gamma \cdot \frac{\tilde{z}(\theta) - \mu}{\sigma + \epsilon} \in \mathbb{R}^{512}, \tag{S73}$$

where $\mu \in \mathbb{R}^{512}$ and $\sigma \in \mathbb{R}^{512}$ are fixed reference statistics and $\epsilon > 0$ is a small stabiliser (added elementwise) to avoid division by zero. In our implementation, (μ, σ) are computed once from an initial set of feature evaluations (e.g., using a fixed seed and the initial θ) and then held fixed for the remainder of training; this ensures that the mapping Φ remains deterministic and does not leak label information.

Interpretation and trade-offs.

The two-marginal design in Eq. (S70)–(S73) offers a practical compromise between expressivity and sample efficiency. A full histogram over $\{0,1\}^{16}$ would require $2^{16} = 65{,}536$ bins and would be extremely sparse for typical shot budgets. In contrast, the two 8-bit marginals use only $2 \times 256 = 512$ bins, yielding more reliable estimates under finite shots while preserving higher-order correlations within each half. The finite-shot concentration behaviour follows the scaling in Supplementary Note S3.3 with effective histogram size $K = 512$.

Implementation note.

In code, if samples are returned as a binary matrix $B \in \{0,1\}^{S\times 16}$, the index mapping in Eq. (S69) can be computed by a dot product with a fixed weight vector $(2^7, 2^6, \ldots, 2^0)$. Histograms are then obtained by counting occurrences over $k \in \{0, \ldots, 255\}$ and normalising by S. The final $z(\theta)$ is produced by concatenation and elementwise standardisation as in Eq. (S73).

S6 Training protocol

This Supplementary Note describes the full training procedure used for PQKD, including teacher optimisation, student optimisation under knowledge distillation, and the update rule for photonic parameters. The protocol is designed to be reproducible and to reflect the operational constraints of a shot-based photonic backend: the photonic module is accessed through repeated sampling to construct features, while the classical student is trained using standard gradient-based optimisation. Because the photonic feature affects the student through a non-analytic sampling interface, we adopt an alternating optimisation strategy that approximates a bilevel objective.

S6.1 Data splits, preprocessing, and evaluation

Unless otherwise stated, we use a fixed train/validation/test split and fixed random seeds for dataset subsampling and mini-batch ordering. Inputs are standardised according to dataset conventions (e.g., per-channel normalisation for CIFAR-10) and augmented only when explicitly indicated. All reported validation and test metrics are computed deterministically (no dropout, fixed batch ordering, no data augmentation at evaluation time). Accuracy is reported as top-1 classification accuracy; cross-entropy refers to the standard negative log-likelihood loss.

S6.2 Teacher training

Teachers are trained with hard-label cross-entropy using Adam with default learning rate 10^{-3} for 100 epochs unless otherwise stated. We use batch size 64 and apply standard regularisation consistent with the teacher architecture (e.g., dropout in `conv3`). Let teacher logits be $t(x)$ and hard labels be y. The teacher objective is

$$\min_{\psi} \; \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}\Big[\,\mathrm{CE}\left(y, \mathrm{softmax}(t(x;\psi))\right)\Big], \tag{S74}$$

where ψ denotes teacher parameters. The best-performing checkpoint is selected using validation accuracy and then evaluated once on the held-out test set.

S6.3 Alternating optimisation for PQKD

Bilevel viewpoint.

PQKD introduces two coupled sets of learnable variables: classical student weights w and photonic circuit parameters θ. The student weights are optimised to minimise a training distillation loss, while θ is optimised to improve generalisation as measured by

a validation proxy objective computed using the photonic feature map. This structure can be expressed as the bilevel problem

$$w^*(\theta) \in \arg\min_{w} \mathscr{L}_{\mathrm{KD}}^{\mathrm{train}}(w, \theta), \qquad \theta^* \in \arg\min_{\theta} \mathscr{L}_{\mathrm{KD}}^{\mathrm{val}}\big(w^*(\theta), \theta\big). \tag{S75}$$

Direct bilevel optimisation is typically computationally expensive, especially when the outer variable θ only influences the model through shot-based samples. We therefore employ an alternating scheme that approximates Eq. (S75) with a small number of outer updates per epoch.

Operational training loop.

Each student epoch consists of two phases:

1. **Photonic phase (outer updates):** perform `theta_updates_per_epoch` $= 10$ SPSA steps to update θ while holding w fixed. Each SPSA step uses a validation mini-batch (or a small fixed set of validation batches) to evaluate the proxy objective $J(\theta)$.
2. **Classical phase (inner updates):** perform one epoch of Adam updates on w using the distillation loss, holding θ fixed (equivalently, holding the photonic feature used by the student fixed for that epoch).

This schedule yields a practical and stable separation of concerns: classical optimisation proceeds with smooth gradients on training data, while photonic optimisation is driven by a generalisation-oriented signal from validation performance and uses only function evaluations through the sampling backend.

S6.4 SPSA photonic updates

Validation proxy objective.

Let $J(\theta)$ denote the scalar objective used for outer-loop updates. In our implementation, $J(\theta)$ is defined as the negative validation accuracy (or, equivalently, the validation distillation loss) computed with the current student weights w and with the photonic feature map evaluated at θ. Because the feature is shot-limited, $J(\theta)$ is stochastic; we reduce variance by averaging over a small number of validation mini-batches.

SPSA estimator.

Simultaneous Perturbation Stochastic Approximation (SPSA) provides an efficient gradient-free update that requires only two objective evaluations per step, independent of $\dim(\theta)$[71]. At SPSA step k, we draw a Rademacher perturbation vector

$$\Delta_k \in \{\pm 1\}^{\dim(\theta)} \qquad \text{with i.i.d. entries.} \tag{S76}$$

We then evaluate the objective at symmetric perturbations

$$\theta_k^{\pm} = \theta_k \pm c\,\Delta_k, \tag{S77}$$

and form the stochastic gradient estimate

$$\widehat{g}_k = \frac{J(\theta_k^+) - J(\theta_k^-)}{2c}\,\Delta_k. \tag{S78}$$

The parameter update is

$$\theta_{k+1} = \theta_k - a\,\widehat{g}_k, \tag{S79}$$

where $a > 0$ is the step size and $c > 0$ is the perturbation scale. For stability and to respect backend parameter constraints, we apply elementwise clipping

$$\theta_{k+1} \leftarrow \mathrm{clip}(\theta_{k+1}; [-\theta_{\max}, \theta_{\max}]). \tag{S80}$$

In all experiments, $(a, c, \theta_{\max})$ are held fixed within a sweep and reported alongside the corresponding results.

Relation to finite-shot noise.

Because $J(\theta)$ depends on shot-limited features (Supplementary Note S3.3), $\widehat{g}_k$ is a noisy estimate. The use of symmetric perturbations in Eq. (S78) cancels some common-mode sampling noise, and evaluating $J(\theta)$ on multiple validation mini-batches further reduces variance. We report noise impact studies by varying the shot budget S and observing its effect on stability and final accuracy (Supplementary Note S6.6).

S6.5 Classical student updates

During the classical phase, student weights w are optimised using Adam on the training distillation objective. Let $s(x; w, \theta)$ denote student logits given the current photonic feature (fixed for the epoch). The update minimises $\mathscr{L}_{\mathrm{KD}}^{\mathrm{train}}(w, \theta)$ defined in Eq. (S2). We use the same batch size (64) as the teacher unless otherwise stated. Training is performed with standard best practices: gradients are backpropagated only through the classical student network; no gradients are propagated through the photonic sampler, consistent with our black-box sampling interface.

S6.6 EMA feature smoothing

Finite-shot photonic sampling produces a stochastic estimate of the conditioning feature $z(\theta)$ used to parameterise the compressed student. If injected directly, this sampling noise appears as high-frequency, epoch-to-epoch innovations in the feature stream and can be amplified by the student's sensitivity to z, degrading optimisation stability at fixed shot budgets (Supplementary Note S3.3). To suppress these fluctuations without modifying the photonic circuit or measurement procedure, we optionally apply exponential moving average (EMA) smoothing to the feature sequence:

$$\bar{z}_t \;=\; \beta\,\bar{z}_{t-1} + (1-\beta)\,z_t(\theta), \qquad \beta \in [0, 1), \tag{S81}$$

where t indexes feature evaluations (epochs in our implementation) and $\bar{z}_0 = z_0(\theta)$. When enabled, the student forward pass uses $\bar{z}_t$ in place of the instantaneous sample $z_t(\theta)$.

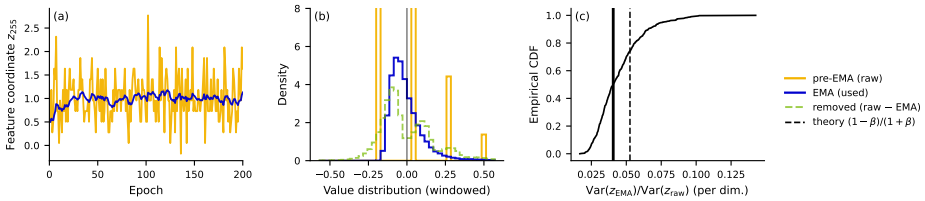

Fig. S1 EMA suppresses shot-noise fluctuations in the photonic conditioning signal. Feature traces recorded during PQKD training at fixed shot budget compare the raw per-epoch photonic feature (pre-EMA) with the EMA-smoothed feature used by the student. **(a)** A representative feature coordinate illustrates attenuation of high-frequency sampling jitter under EMA. **(b)** Windowed value distributions across feature dimensions for the raw and EMA-used signals; the dashed curve indicates the removed residual ($z_{\mathrm{raw}} - z_{\mathrm{EMA}}$). **(c)** Empirical CDF of the per-dimension variance ratio $\mathrm{Var}(z_{\mathrm{EMA}})/\mathrm{Var}(z_{\mathrm{raw}})$ with the expected attenuation level shown for reference.

EMA acts as a causal low-pass filter: it preserves slowly varying components of the photonic signal while attenuating rapid innovations that are predominantly induced by finite-shot fluctuations. A useful interpretation follows from the decomposition $z_t(\theta) = \mu(\theta) + \varepsilon_t$ with $\mathbb{E}[\varepsilon_t] = 0$ and approximately independent increments ε_t across successive evaluations. Under this standard approximation, $\mathbb{E}[\bar{z}_t] = \mu(\theta)$ (no bias in the mean signal), and the variance of the smoothed estimate is reduced relative to the raw estimate by

$$\frac{\mathrm{Var}(\bar{z}_t)}{\mathrm{Var}(z_t)} \approx \frac{1-\beta}{1+\beta}, \tag{S82}$$

with the reduction becoming sharper as $\beta \to 1$. Equivalently, EMA yields an effective increase in sampling budget,

$$S_{\mathrm{eff}} \approx \frac{1+\beta}{1-\beta} S, \tag{S83}$$

so that the student experiences a feature stream comparable to one obtained with substantially more shots, but at the same physical measurement cost per epoch.

In our MNIST EMA-mechanism trace at S=200 shots and β=0.9, the distribution of per-dimension variance ratios $\mathrm{Var}(z_{\mathrm{used}})/\mathrm{Var}(z_{\mathrm{raw}})$ concentrates well below unity, with a median 0.0406 (IQR 0.0322–0.0532) over the analysis window, consistent with the theoretical attenuation level $(1-\beta)/(1+\beta) = 0.0526$ and the corresponding amplification $S_{\mathrm{eff}} \approx 19\,S$. These observations support the role of EMA as a simple, hardware-compatible variance-reduction mechanism for stabilising PQKD in shot-limited regimes, and complement the robustness results in Sec. 2.4.

S6.7 Algorithms

Algorithms S1–S3 summarise the full PQKD training loop, the SPSA outer-loop update, and the photonic feature extraction used in the main experiments ($N = 16$, $d = 512$). These procedures directly implement the alternating optimisation described in Eq. (S75) and Eq. (S79).

Algorithm S1 PQKD training loop (teacher → student + photonic updates)

Require: Data loaders $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$, teacher f_T, student f_S, photonic sampler Sampler, KD hyperparameters (τ, λ), shots S, outer steps U_{out}, student epochs E_{student}

Train teacher f_T with cross-entropy on $\mathcal{D}_{\text{train}}$; freeze f_T
Initialise photonic parameters $\theta \leftarrow 0$; initialise student parameters w
Initialise (optional) EMA state $\bar{z} \leftarrow \emptyset$
for epoch $= 1$ to E_{student} **do** ▷ **Photonic phase: update θ with w fixed**
  for $u = 1$ to U_{out} **do**
    $\theta \leftarrow \text{SPSAUpdate}(\theta, w, f_T, f_S, \mathcal{D}_{\text{val}}, \text{Sampler}, S, \tau, \lambda)$ ▷ Algorithm S2
  end for ▷ **Classical phase: update w with θ fixed**
  $z(\theta) \leftarrow \text{PhotonicFeature}(\text{Sampler}, \theta, S)$ ▷ Algorithm S3
  if EMA enabled **then**
    $\bar{z} \leftarrow \beta\bar{z} + (1-\beta)z(\theta)$ (initialise $\bar{z} = z(\theta)$ if empty)
    Use $\bar{z}$ as the conditioning feature for this epoch
  else
    Use $z(\theta)$ as the conditioning feature for this epoch
  end if
  Train student for one epoch with Adam on $\mathcal{L}_{\text{KD}}(w, \theta)$ using $\mathcal{D}_{\text{train}}$
end for

Algorithm S2 SPSA update for photonic parameters

Require: Current θ, fixed student weights w, step sizes (a, c), clipping bound $\theta_{\max}$, validation objective $J(\theta)$ computed from KD loss or $-$ accuracy on $\mathcal{D}_{\text{val}}$

Sample $\Delta \in \{\pm 1\}^{\dim(\theta)}$
$\theta^+ \leftarrow \text{clip}(\theta + c\Delta,\ -\theta_{\max}, \theta_{\max})$
$\theta^- \leftarrow \text{clip}(\theta - c\Delta,\ -\theta_{\max}, \theta_{\max})$
$J^+ \leftarrow J(\theta^+)$ ▷ evaluate using photonic feature extraction + KD forward on $\mathcal{D}_{\text{val}}$
$J^- \leftarrow J(\theta^-)$
$\widehat{g} \leftarrow \dfrac{J^+ - J^-}{2c}\Delta$
$\theta \leftarrow \text{clip}(\theta - a\widehat{g},\ -\theta_{\max}, \theta_{\max})$
return θ

Algorithm S3 Photonic feature extraction ($N = 16$, $d = 512$)

Require: Photonic sampler Sampler, parameters θ, shots S, fixed standardisation (μ, σ, ϵ), scaling γ

Sample S outcomes over $N = 16$ modes; threshold to bits $b_s \in \{0,1\}^{16}$
Split each b_s into two halves $b_s^{(1)}, b_s^{(2)} \in \{0,1\}^8$; map each half to idx $\in \{0, \ldots, 255\}$
Form two normalised histograms $h^{(1)}, h^{(2)} \in \mathbb{R}^{256}$ and concatenate $\tilde{z} \leftarrow [h^{(1)}; h^{(2)}] \in \mathbb{R}^{512}$
Standardise and scale: $z(\theta) \leftarrow \gamma \cdot \dfrac{\tilde{z} - \mu}{\sigma + \epsilon}$
return $z(\theta)$