CONQUER: MODULAR CONDITIONAL CONTROL OF FROZEN IQP GENERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained instantaneous quantum polynomial (IQP) generators learn data distributions but lack direct control over the generated samples when trained unconditionally. Practical applications often require generation conditioned on a specific mode or class. We study how to add this capability without updating the pretrained backbone or departing from the IQP circuit family. We propose ConQuER, a modular post-training framework that equips frozen IQP generators with a reusable conditional generation interface. Its controller adopts the same IQP structure and composes directly with the backbone to steer generation toward requested conditions. On Binary Blobs, a 784-support controller achieves mean conditional squared maximum mean discrepancy (MMD^2) of 0.0105 and a mean requested-mode hit rate of 78.49%. On 4×4 binarized MNIST-3/5, a controller with 1,380 trainable parameters serves both classes and reduces mean conditional MMD^2 by 94.1% relative to the frozen unconditional backbone. Further evaluations on these tasks and Shapes20 characterize the trade-off between conditional distribution quality and parameter budget and demonstrate parameter savings through shared control across conditions.

1 INTRODUCTION

Quantum generative modeling uses quantum systems to learn data distributions and produce new samples (Gao et al., 2018). Quantum circuit Born machines implement this idea through parameterized circuits whose measurement probabilities define a distribution over bitstrings (Liu & Wang, 2018; Benedetti et al., 2019; Zhu et al., 2019). Instantaneous quantum polynomial (IQP) circuits provide a structured family of such generators, consisting of commuting phase gates between Hadamard layers (Shepherd & Bremner, 2009). Despite this restricted structure, IQP circuit families support sampling tasks believed to be classically hard under standard complexity assumptions (Bremner et al., 2011; 2017). At the same time, suitable maximum mean discrepancy (MMD) objectives and their gradients admit efficient classical estimators that do not require full state-vector simulation (Recio-Armengol et al., 2026; Recio-Armengol & Bowles, 2025). These methods have enabled generative learning with IQP models containing up to a thousand qubits, establishing a concrete basis for studying the capabilities and uses of trained IQP generators (Recio-Armengol et al., 2026).

Using a pretrained IQP generator may require samples from a particular mode or class, rather than from the overall distribution it has learned. For example, a model trained on several digit classes may be asked to generate one specified class while retaining the variation within that class. Prior work has studied conditional quantum generation through conditional quantum circuit Born machines (Zeng et al., 2023) and hybrid quantum–classical conditional generative adversarial networks (Liu et al., 2021). However, an unconditional IQP generator has no input through which a requested class can influence generation. Even an accurate model of the pooled distribution therefore requires an additional mechanism to make its samples responsive to such requests.

Postselection obtains condition-specific samples by applying an external acceptance rule to an unconditional generator. The request determines which measured outputs are retained, while the source distribution remains unchanged. Alternatively, training or fully fine-tuning a separate model for each condition provides greater freedom to fit the target distributions, but requires maintaining separate full-model parameter sets. We investigate a different approach: retaining the pretrained generator and learning an additional control module that makes its output depend on the requested

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

condition. The backbone parameters remain fixed throughout adaptation, while the control module is trained on condition-labeled data.

An IQP-based control module offers a natural way to integrate this additional capability with the existing generator. Because the controller and backbone belong to the same circuit family, their composition can be expressed as a single IQP circuit without intermediate measurement or re-encoding between the two modules. The combined model also remains compatible with the classical estimators available for suitable MMD objectives and their gradients (Recio-Armengol et al., 2026; Recio-Armengol & Bowles, 2025). The remaining challenge is to select and learn phase adjustments that approximate the requested distributions within a limited control budget.

We propose ConQuER, a modular post-training framework that equips frozen IQP generators with a conditional generation interface. Given a requested mode or class, its controller supplies phase offsets on a selected set of qubit supports. Adjacent Hadamard layers cancel, and commuting phase gates acting on the same support combine through angle addition, yielding condition-dependent effective phases while leaving the stored backbone parameters unchanged. The controller is optimized using conditional MMD^2 to steer the sampling distribution toward the requested target before measurement. We evaluate ConQuER on Binary Blobs, 4×4 binarized MNIST-3/5, and Shapes20, demonstrating conditional generation from pretrained backbones without updating their original parameters. Budget sweeps and comparisons with separately trained per-condition generators further characterize the trade-off between conditional distribution quality and trainable parameter count.

The contributions of this work are summarized as follows.

- **A reusable conditional interface for frozen IQP generators.** We introduce ConQuER, a modular post-training framework that maps requested modes to learned phase offsets on selected supports while keeping the stored backbone parameters fixed. The resulting conditional generator remains IQP, preserving compatibility with existing classical estimators of suitable MMD objectives and their gradients.
- **Budgeted and extensible phase control.** We develop support-selection and parameter-sharing mechanisms to allocate conditional capacity and reuse learned control across conditions. In a fixed-support leave-one-condition-out study on Binary Blobs, adapting each held-out condition with 64 new parameters achieves comparable mean conditional MMD^2 to independent 784-parameter adaptation, reducing the marginal parameter requirement by 91.8% without modifying existing condition models.
- **Quantitative evidence of conditional generation and parameter efficiency.** On Binary Blobs, ConQuER achieves mean conditional MMD^2 of 0.0105 and a mean requested-mode hit rate of 78.49% using 784 supports. On 4×4 binarized MNIST-3/5, a 1,380-parameter controller serves both classes and reduces mean conditional MMD^2 by 94.1% relative to the frozen unconditional backbone. Matched comparisons, ablations, and budget sweeps characterize quality–parameter trade-offs.

2 Background and Problem Formulation

2.1 Background: IQP Generators

An n-qubit IQP generator consists of commuting diagonal phase gates between Hadamard layers (Shepherd & Bremner, 2009). Let $\mathcal{G}$ be a family of nonempty qubit supports $S \subseteq \{1, \ldots, n\}$. For parameters $\theta \in \mathbb{R}^{|\mathcal{G}|}$, measurement in the computational basis defines the Born distribution

$$p_\theta(x) = \left| \langle x | H^{\otimes n} \left(\prod_{S \in \mathcal{G}} \exp(-i\theta_S Z_S) \right) H^{\otimes n} |0^n\rangle \right|^2, \qquad x \in \{0,1\}^n, \tag{1}$$

where Z_S acts as Pauli Z on the qubits in S and as the identity elsewhere. The support family $\mathcal{G}$ specifies the circuit topology and interaction orders. For suitable kernels, maximum mean discrepancy (MMD) objectives and their gradients admit classical estimators that exploit the IQP structure without full state-vector simulation (Recio-Armengol et al., 2026; Recio-Armengol & Bowles, 2025).

2.2 PROBLEM: POST-TRAINING CONDITIONAL GENERATION

Let $m \in \{1, \dots, C\}$ denote a discrete condition, such as a mixture component or a class label. A pretrained unconditional IQP generator $p_{\theta_{\text{base}}}$ approximates the marginal data distribution

$$p_{\text{data}}(x) = \sum_{m=1}^{C} \pi_m \, p_{\text{data}}(x \mid m), \qquad \pi_m \geq 0, \quad \sum_{m=1}^{C} \pi_m = 1. \tag{2}$$

Fitting this marginal does not provide an input through which a requested condition can influence generation.

During adaptation, we assume access to condition-labeled training data and learn controller parameters ψ while keeping θ_{base} fixed. For each request m, the goal is to obtain

$$p_{\theta_{\text{base}},\psi}(x \mid m) \approx p_{\text{data}}(x \mid m). \tag{3}$$

This requires matching the conditional distribution, including its within-condition variation. The controller must act before measurement, preserve the IQP circuit family, and operate within a specified budget of additional trainable parameters. We target direct conditional generation: only the requested condition is supplied, and every measurement outcome is returned without an external acceptance rule.

3 CONQUER: MODULAR CONDITIONAL PHASE CONTROL

Figure 1 summarizes how ConQuER equips a frozen unconditional IQP generator with a conditional sampling interface. Unlike the request-agnostic backbone on the left, the composed model maps a condition m to controller phase offsets that modify the sampling distribution before measurement. During adaptation, condition-labeled data supervise the composed generator, while only the controller parameters are optimized. Once adapted, a request configures the controller phases without further training; the comparison on the right illustrates the intended change from unconditional to conditional outputs.

3.1 IQP-COMPATIBLE CONDITIONAL PHASE CONTROL

Let $U_{\text{base}}(\theta_{\text{base}})$ denote the pretrained IQP unitary underlying Eq. (1), with support family $\mathcal{G}_{\text{base}}$. We introduce a controller on the same n qubits with support set $\mathcal{M} = \{S_1, \dots, S_K\}$, where $K = |\mathcal{M}|$. For a requested condition m, a classical parameter map produces the phase offsets $\phi_m = f_\psi(m) \in \mathbb{R}^K$. The controller follows the standard IQP structure of commuting diagonal phase gates between Hadamard layers (Shepherd & Bremner, 2009):

$$U_{\text{ctrl}}(m) = H^{\otimes n} \left(\prod_{r=1}^{K} \exp(-i\phi_{m,r} Z_{S_r}) \right) H^{\otimes n}. \tag{4}$$

Here, ϕ_m denotes the applied phase offsets, including any prescribed controller output scale in f_ψ. The support set specifies the available phase gates, whereas f_ψ determines their condition-dependent parameters.

We compose the controller directly with the frozen backbone:

$$\begin{aligned} U_{\text{cond}}(m) &= U_{\text{base}}(\theta_{\text{base}}) U_{\text{ctrl}}(m) \\ &= H^{\otimes n} \left(\prod_{S \in \mathcal{G}_{\text{all}}} \exp(-i\theta_{\text{eff},S}(m) Z_S) \right) H^{\otimes n}, \end{aligned} \tag{5}$$

where $\mathcal{G}_{\text{all}} = \mathcal{G}_{\text{base}} \cup \mathcal{M}$. This equality follows because the adjacent Hadamard layers cancel and commuting phase gates on identical supports combine through angle addition. Writing $\bar{\theta}_{\text{base}}$ for the backbone phase vector extended by zeros to $\mathcal{G}_{\text{all}}$, the effective phases are

$$\theta_{\text{eff}}(m) = \bar{\theta}_{\text{base}} + \text{Scatter}_{\mathcal{M}}(f_\psi(m)) \, . \tag{6}$$

The scatter operator places $\phi_{m,r}$ at coordinate S_r and assigns zero to coordinates outside $\mathcal{M}$. Appendix A.1 provides the algebraic derivation.

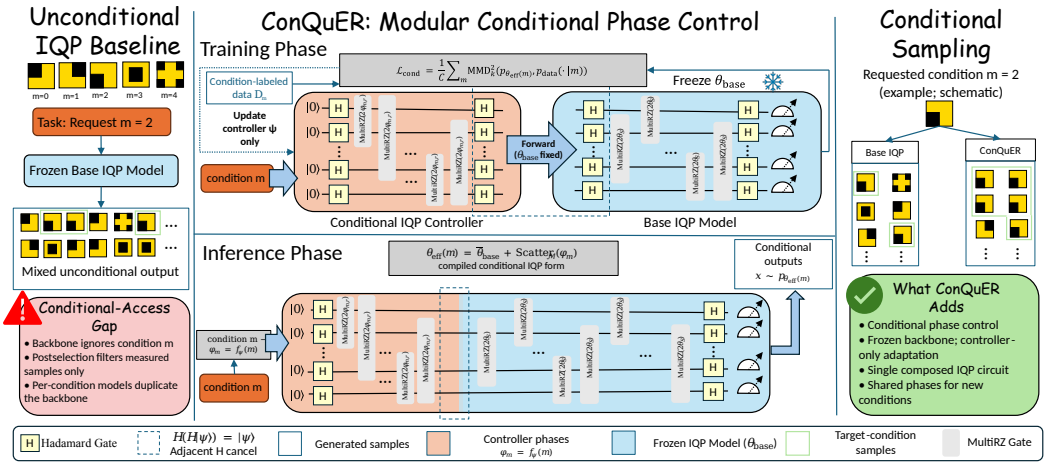

Figure 1: Overview of ConQuER. **Left:** the frozen unconditional backbone produces the same distribution regardless of the requested condition. **Upper center:** condition-labeled data supervise the conditional MMD^2 objective of the composed generator, with updates restricted to controller parameters. **Lower center:** a request determines phase offsets $\phi_m = f_\psi(m)$. Adjacent Hadamard cancellation and addition of phases on identical supports yield a single IQP circuit with effective phases $\theta_{\mathrm{eff}}(m) = \bar{\theta}_{\mathrm{base}} + \mathrm{Scatter}_{\mathcal{M}}(\phi_m)$. **Right:** schematic output examples compare the backbone and the adapted generator under the same request.

Computational-basis measurement therefore yields $p_{\theta_{\mathrm{base}},\psi}(x \mid m) = p_{\theta_{\mathrm{eff}}(m)}(x)$. Only ψ is optimized during adaptation: the backbone remains a fixed reference, while the effective phases vary with the requested condition. Setting $\phi_m = 0$ recovers the original unconditional generator exactly. Conditioning thus changes the sampling distribution before measurement, without intermediate measurement or re-encoding between modules. Because the composed circuit remains IQP, it retains compatibility with existing classical estimators for suitable MMD objectives and their gradients (Recio-Armengol et al., 2026; Recio-Armengol & Bowles, 2025).

Equation (6) expresses control as condition-dependent additive phase updates. When $\mathcal{M} \subseteq \mathcal{G}_{\mathrm{base}}$, this is algebraically equivalent to sparse phase fine-tuning with the parameterization specified by f_ψ. Here, $\mathcal{M}$ determines which phase coordinates can change, while f_ψ determines the number of trainable parameters across conditions. Thus, the support count K need not equal the total controller parameter count.

3.2 Learning the Conditional Controller

The upper training loop in Fig. 1 learns the controller through the output distribution of the composed generator. For each condition m, the request determines both the applied phase vector and the training subset $\mathcal{D}_m$ used as its target. For C training conditions and a given support set $\mathcal{M}$, we minimize the average conditional squared MMD (Gretton et al., 2012):

$$\mathcal{L}_{\mathrm{cond}}(\psi; \mathcal{M}) = \frac{1}{C} \sum_{m=1}^{C} \widehat{\mathrm{MMD}}_k^2 \left(p_{\theta_{\mathrm{eff}}(m)}, p_{\mathrm{data}}(\cdot \mid m) \right), \tag{7}$$

where target-distribution terms are estimated using $\mathcal{D}_m$. We use Gaussian kernels $k_\sigma(x, x') = \exp(-\|x - x'\|_2^2/(2\sigma^2))$, averaging the corresponding losses when multiple bandwidths are used.

Gradients propagate through the effective phases and the map f_ψ to update only ψ, while θ_{base} remains fixed.

Our configurations use either a fixed support set or offline support growth. The MNIST-3/5 configuration includes all one- and two-qubit supports. For support growth, let $g_{m,S}$ be the derivative of condition m's training loss with respect to the phase on candidate support S, evaluated at the current effective phases. We rank unused supports by $\left(C^{-1} \sum_m g_{m,S}^2\right)^{1/2}$, add the highest-scoring

candidates, and optimize conditional phases on the enlarged set. When candidate gradients vanish numerically, a temporary phase perturbation is used for scoring. At later growth stages, new offsets start at zero, while existing offsets and optimizer state are retained. Support construction is completed during adaptation; request-time generation uses the resulting fixed topology.

On these supports, a direct phase map stores a separate K-dimensional vector for each condition, requiring CK trainable parameters. A factorized alternative uses trainable condition embeddings and a common decoder:

$$\phi_m = W^\top e_m, \qquad E \in \mathbb{R}^{C\times d}, \qquad W \in \mathbb{R}^{d\times K}, \tag{8}$$

where row m of E is $e_m^\top$ and $\psi = \{E, W\}$. This representation shares a decoder and requires $d(C + K)$ trainable parameters.

3.3 Reusing and extending the conditional interface

To reuse phase values across requests, we introduce coordinate-wise sharing within f_ψ, while retaining the same circuit composition. For a fixed support set, we partition $\mathcal{M} = \mathcal{M}_{\text{shared}} \dot\cup \mathcal{M}_{\text{private}}$, with $|\mathcal{M}_{\text{private}}| = q$. Shared coordinates use a common phase value a_r, while private coordinates use a condition-specific value $b_{m,r}$:

$$\phi_{m,r} = \begin{cases} a_r, & S_r \in \mathcal{M}_{\text{shared}}, \\ b_{m,r}, & S_r \in \mathcal{M}_{\text{private}}. \end{cases} \tag{9}$$

In the gradient-selected configuration, let $g_{m,r}$ denote the conditional-loss gradient on coordinate r at zero controller offsets. We select the q coordinates with the largest $\sum_m (g_{m,r} - \bar g_r)^2$, where $\bar g_r$ is the mean over the conditions available for shared-controller training. This criterion allocates private values to coordinates whose gradients differ across conditions within the fixed $\mathcal{M}$. The offset-selected variant replaces these gradients with fitted condition-specific offsets in the same score. After selecting the partition, we initialize a new random phase table, averaging its shared columns across conditions and retaining its condition-specific entries on private columns.

Once the partition is fixed, the shared values and the private values of the available conditions are jointly optimized using Eq. (7). Shared parameters receive gradient contributions from all training conditions, whereas private parameters receive contributions only from their associated condition. The resulting controller contains

$$P_{\text{ctrl}}(C) = (K - q) + Cq \tag{10}$$

independent trainable parameters, excluding the frozen backbone. Setting $q = K$ recovers the independent phase table; setting $q = 0$ makes the offsets identical across conditions. For intermediate values, each request still uses K phase offsets, but only q values are specific to that condition.

This separation also allows the learned shared values to be reused when adding a condition. After training on conditions $1, \dots, C$, we freeze the backbone, the support set and partition, the shared values, and all existing private values. For condition $C + 1$, we introduce $b_{C+1} \in \mathbb{R}^q$ on $\mathcal{M}_{\text{private}}$ and combine it with the frozen shared values according to Eq. (9). Using $\mathcal{D}_{C+1}$, we optimize only this new vector by minimizing the corresponding conditional MMD^2 term. The marginal parameter requirement is

$$\Delta P_{\text{ctrl}} = P_{\text{ctrl}}(C + 1) - P_{\text{ctrl}}(C) = q. \tag{11}$$

Since neither the shared values nor the existing private values change, the effective phases of every previously adapted condition remain identical. Their output distributions are therefore preserved.

4 ConQuER: Evaluation

We study conditional generation on Binary Blobs, MNIST-3/5, and Shapes20, relating the quality obtained from frozen IQP backbones to controller support allocation and parameter sharing. We further examine whether learned shared control can be reused when adding a condition.

4.1 EXPERIMENTAL SETUP

Datasets. Binary Blobs contains eight structured 16-bit modes and follows the benchmark used in prior IQP generative modeling (Bowles et al., 2024; Recio-Armengol et al., 2026). For MNIST-3/5 (Lecun et al., 1998), we retain digits 3 and 5 from the official split, resize them to 4×4 using fixed non-antialiased bilinear interpolation, and binarize normalized intensities at 0.5. Shapes20 contains 20-bit (4×5) patterns from four classes, H/I/T/X, with nine variants per class formed by translations and full-row occlusion, followed by 6% bit noise.

Controller configurations. The support count $K = |\mathcal{M}|$ specifies the number of selected qubit subsets, each receiving one phase offset per request. Our main Blobs and Shapes20 controllers use separately constructed 784-support sets obtained through offline growth along $256 \to 512 \to 784$, with 120 optimization steps per stage. Blobs considers all nonempty supports, while Shapes20 considers supports of orders one through six. An independent phase vector for each condition gives CK trainable parameters for C conditions: 6,272 on Blobs and 3,136 on Shapes20. MNIST uses all 136 one- and two-qubit supports and the factorized phase map of Section 3.2, with $d = 10$ and 1,380 trainable parameters. Controllers are trained on condition-labeled data while the pooled-data IQP backbones remain frozen.

Implementation. All experiments use CPU-only computation on a dual-socket Intel Xeon Gold 6442Y server with 512 GiB RAM, JAX/jaxlib 0.4.28, and double precision. Blobs uses a custom Born/Walsh–Hadamard simulator; MNIST uses iqpopt 2024.7.0 with PennyLane-Lightning 0.41.1 (Bergholm et al., 2022). Backbone preparation uses Hellinger amplitude projection on Blobs and stochastic MMD optimization on MNIST. The MNIST output-scale schedule is given in Appendix A.2. Controller configurations and checkpoints are selected using validation conditional MMD^2.

Metrics. The primary metric is condition-averaged squared MMD (Gretton et al., 2012). For all three tasks, we use the fixed kernel $k = \frac{1}{2}(k_{0.6} + k_{1.3})$, where $k_\sigma(x, y) = \exp[-d_H(x, y)/(2\sigma^2)]$ and d_H is Hamming distance. Blobs and Shapes20 use complete model probability distributions against empirical test distributions in V form; MNIST direct generation uses a two-sample U-statistic with 512 generated and 512 test examples per digit under fixed evaluation seeds. Secondary metrics are nearest-prototype hit on Blobs, classifier-based requested-digit consistency on MNIST, and classifier-based target hit on Shapes20. The MNIST RBF-SVM and Shapes20 logistic-regression classifiers are fitted on training data; both classifier-based metrics use complete model probabilities. The MNIST evaluator attains 75.82% balanced accuracy on real test images; conditional MMD^2 and supplementary empirical total variation (TV) assess distributional quality independently of this classifier. Postselection uses training-fitted logistic classifiers on Blobs and Shapes20 and the MNIST SVM (Appendix C.3). Full protocols and additional metrics are provided in Appendix B.

4.2 OVERALL CONDITIONAL GENERATION

On Blobs, the mode-assignment matrix changes from identical rows for the frozen generator to a diagonally concentrated pattern under ConQuER (Fig. 2(a)). Mean requested-mode hit increases from 12.50% to 78.49%, averaged over eight modes and three backbones, with conditional $\mathrm{MMD}^2 = 0.010494$ (Table 1). Changing the request therefore redirects the same pretrained generator toward different components of the pooled distribution.

Class-dependent generation also emerges on MNIST (Fig. 2(b)). Mean conditional MMD^2 decreases from 0.038339 to 0.00227, a 94.1% reduction, using one controller for both digits. Independent per-digit IQPs attain 0.00128 with 29,784 trainable parameters, compared with 1,380 additional parameters for ConQuER. The pixel marginals and distributional results together show conditional control with a generic low-order support set.

On Shapes20, ConQuER (fixed supports) fits independent class offsets on 196 supports selected once from pooled training data. The growth-based configuration reduces MMD^2 from 0.04112 to 0.02416 and increases target hit from 53.87% to 64.47%. Full fine-tuning of four condition-specific IQPs attains 0.00245 using $4 \times 60{,}459 = 241{,}836$ trainable parameters, compared with 3,136 for ConQuER. The conditional autoregressive (AR) reference uses logistic conditionals with 290 parameters and attains 0.01233. ConQuER improves over the fixed-support controller while retaining the frozen

Table 1: **Conditional quality, trainable parameters, and sampling cost.** K counts controller supports; trainable counts exclude frozen backbone parameters and acceptance classifiers. Blobs reports means over three backbones using controller initialization seed 0. MNIST trained-model entries report mean $\pm$ sample standard deviation over five seeds, with conditions averaged within each run; the frozen reference is one checkpoint. Shapes20 controller entries average three runs; full fine-tuning and AR each use one run. $\overline{N}_{\mathrm{raw}}$ denotes expected raw draws per returned sample. † marks classifier-assisted postselection at request time; unmarked rows return every draw. MNIST and Shapes20 postselection scores are 100% by construction.

Method	Supports K	Trainable	$\mathrm{MMD}^2 \downarrow$	Control metric (%) $\uparrow$	$\overline{N}_{\mathrm{raw}} \downarrow$
Binary Blobs				*Mode hit*	
Frozen + postselection†	—	0	0.00420	96.23	8.02
ConQuER	784	6,272	0.010494	78.49	1
MNIST-3/5				*Consistency*	
Frozen IQP	—	0	0.038339	50.00	1
Frozen + postselection†	—	0	0.01761	100	2.07
ConQuER	136	1,380	0.00227 ± 0.00020	70.39 ± 0.25	1
Two independent IQPs	—	29,784	0.00128 ± 0.00014	73.45 ± 0.21	1
Shapes20				*Target hit*	
Frozen + postselection†	—	0	0.00729	100	4.01
ConQuER (fixed supports)	196	784	0.04112	53.87	1
ConQuER	784	3,136	0.02416	64.47	1
Full fine-tuning (IQP)	—	241,836	0.00245	84.77	1
Conditional AR (classical)	—	290	0.01233	89.80	1

backbone, but the classical AR reference achieves a closer fit than ConQuER with fewer trainable parameters.

Table 1 distinguishes direct sampling from classifier-assisted postselection. Frozen-backbone filtering achieves lower conditional MMD^2 than direct ConQuER on Blobs and Shapes20, but higher error on MNIST. Using ConQuER as the postselection source reduces expected raw draws per accepted sample from 8.02 to 1.28 on Blobs and from 4.01 to 1.56 on Shapes20, but yields higher MMD^2 and TV than frozen-backbone filtering (Appendix C.3).

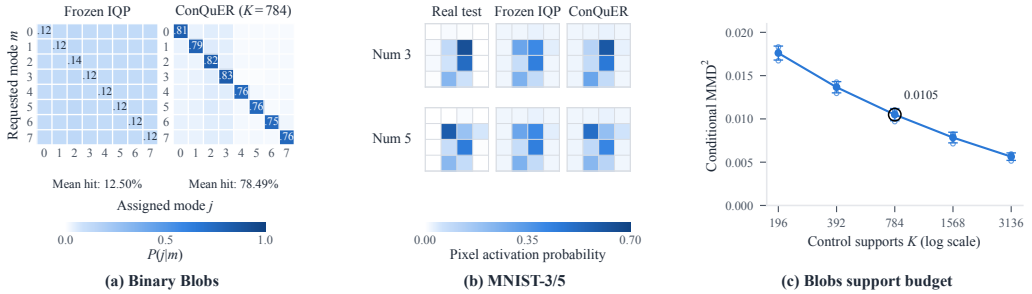

Figure 2: **Conditional generation and support budget.** (a) Blobs mode-assignment probabilities $P(j \mid m)$. (b) MNIST pixel-activation probabilities for requests 3 and 5 ($K = 136$ for ConQuER). (c) Blobs conditional MMD^2 versus support count. Hollow markers show condition-averaged results for each backbone; filled markers and error bars show their mean $\pm$ standard deviation across three backbones. The black ring marks the main $K = 784$ configuration.

4.3 Support Design and Budget

Support allocation affects the quality obtained at a fixed final support count. On Blobs backbone 0 at $K = 784$, growth achieves mean conditional $\mathrm{MMD}^2 = 0.009776$ and 79.16% hit, compared with 0.013904 and 75.12% for one-shot selection, and 0.261071 and 36.02% for random supports.

Table 2: **New-condition adaptation on Binary Blobs.** Mean held-out-condition test MMD^2 over 24 paired experiments: eight conditions on each of three backbones, with one initialization per configuration. All methods use the existing 784-support set; trained variants use 360 new-condition optimization steps. Shared-controller references use the corresponding seven-condition Stage-A model with $q = 64$. The last column counts pairs with lower error than the independent cold start.

New-condition adaptation	New params	$\mathrm{MMD}^2 \downarrow$	Lower / 24
Shared, existing private coordinates	64	0.01106	19
Independent cold start	784	0.01156	—
Shared-mean warm start	784	0.01181	10
New-condition-selected coordinates	64	0.02610	5
Shared mean, no training	0	0.41847	0

Random supports match growth's interaction-order counts at each stage. Replaying the support schedule discovered by growth seed 0, retraining with zero backbone phases yields higher mode hit (81.70% versus 79.16%) but worse conditional MMD^2 (0.013049 versus 0.009776). Thus, retaining the pretrained phases gives a closer conditional fit in this comparison, despite the lower mode hit. Growth, one-shot, and backbone removal average three controller seeds; random supports average five topology seeds with controller seed 0 (Appendix C).

A separate three-backbone study on Shapes20 examines training on the final support sets already discovered by growth. Optimizing all offsets on these fixed dictionaries from the outset gives mean $\mathrm{MMD}^2 = 0.02633$, compared with 0.02664 for dynamic growth. The discovered supports therefore remain effective when reused as fixed control dictionaries.

Increasing the number of supports further improves conditional quality on Blobs. Figure 2(c) spans $K \in \{196, 392, 784, 1568, 3136\}$. From 784 to 3136 supports, mean MMD^2 decreases from 0.010494 to 0.005639 and hit increases from 78.49% to 84.24%. The corresponding independent phase tables grow from 6,272 to 25,088 parameters. Additional control directions improve the fit, but storing separate values for every condition makes the parameter requirement grow with both support count and the number of conditions.

4.4 Parameter sharing and interface extension

Phase sharing reduces this storage requirement while retaining the selected support sets. On Blobs, gradient-selected sharing assigns 64 private coordinates to each condition and shares the remaining 720 values. The eight-condition controller stores $720 + 8 \times 64 = 1{,}232$ parameters, 80.4% fewer than the independent phase table. In a separate comparison across three backbones with controller seed 0, both variants are reinitialized and trained on the same full growth-derived dictionary for each backbone, rather than continued from the growth checkpoints. Under this fixed-dictionary retraining protocol, shared control achieves mean conditional $\mathrm{MMD}^2 = 0.009838$ and 78.51% hit, compared with 0.014669 and 74.99% for independent offsets (Appendix D).

On Shapes20, offset-selected sharing with 256 private coordinates per condition reduces the controller from 3,136 to 1,552 parameters. In a two-seed retraining comparison, mean MMD^2 changes from 0.02406 for independent offsets to 0.02505, exchanging a 4.1% increase in error for 50.5% fewer stored control parameters.

The shared/private structure also supports adding requests to an existing interface. The Blobs leave-one-condition-out study reuses the pooled-data backbone and a support dictionary discovered using all eight conditions. For each holdout, Stage A selects private coordinates and trains the shared/private controller using the other seven conditions. Stage B freezes all existing parameters and fits a new 64-dimensional private vector, initialized from the mean of the seven Stage-A private rows, using labeled data from the held-out condition. The independent reference fits 784 new offsets on the same backbone and supports. The shared-mean references use the mean of the seven Stage-A offset rows. The new-condition-selected reference starts from this mean row and trains the 64 coordinates with the largest absolute new-condition gradients, keeping the remaining entries fixed.

Across eight held-out conditions and three backbones, shared adaptation achieves mean $\mathrm{MMD}^2 = 0.01106$, compared with 0.01156 for independent adaptation, and yields lower error in 19 of 24

pairs (Table 2). The mean errors are close despite a 91.8% reduction in newly learned parameters. Reusing the existing private coordinates also yields lower mean error than the full-vector shared-mean warm start or the evaluated new-condition coordinate-selection rule. An additional request is thus accommodated through a small private vector, while existing requests retain their original effective phases and output distributions.

5 Related Work

Quantum generative modeling. Quantum generative models include circuit Born machines (Liu & Wang, 2018; Benedetti et al., 2019), adversarial networks (Dallaire-Demers & Killoran, 2018), and Ising Born machines linking training to sampling complexity (Coyle et al., 2020). MMD objectives are used to train classical moment-matching networks (Li et al., 2015) and quantum Born machines, with kernel-dependent trainability and sensitivity to higher-order correlations (Rudolph et al., 2024). ConQuER builds on classical IQP MMD estimators (Recio-Armengol et al., 2026) and IQPopt (Recio-Armengol & Bowles, 2025).

Conditional generation and control. Conditional Born machines encode labels or control parameters (Zeng et al., 2023; Kiss et al., 2022); hybrid conditional GANs use quantum generators and classical discriminators (Liu et al., 2021). ControlNet (Zhang et al., 2023) and T2I-Adapter (Mou et al., 2023) learn neural conditioning modules for frozen diffusion models. ConQuER follows the same post-training control principle, but operates on commuting phase coordinates: controller offsets add to backbone phases to produce a single condition-dependent IQP circuit.

Adaptive circuit construction. ADAPT-VQE (Grimsley et al., 2019) and qubit-ADAPT-VQE (Tang et al., 2021) alternate gradient-based operator selection with parameter optimization to construct ansätze for ground-state preparation. ConQuER uses conditional MMD gradients aggregated across requests to grow a common dictionary of commuting phase supports, with condition-specific offsets optimized on the enlarged dictionary.

Parameter-efficient adaptation and reuse. Adapters (Houlsby et al., 2019), LoRA (Hu et al., 2022), and Diff Pruning (Guo et al., 2021) adapt frozen models through modules, low-rank updates, and sparse parameter differences, respectively. Piggyback (Mallya et al., 2018) learns task-specific masks; AdapterFusion (Pfeiffer et al., 2021) combines pretrained adapters. ConQuER shares phase values across requests and fits private values for new conditions without changing existing models.

6 Limitations and Future Work

ConQuER allocates control capacity through a prescribed support budget and gradient-based growth within a predefined candidate pool. The growth procedure retains previously selected supports, motivating support replacement as an alternative to increasing the budget. Phase sharing uses a fixed shared/private partition, and its quality trade-off varies across tasks. Learning this partition during controller optimization, or sharing phases within groups of conditions, could provide more flexible reuse. Interface extension fits new private values while keeping the control dictionary and shared phases fixed, restricting adaptation to previously allocated private coordinates. Extending the dictionary or sharing structure while preserving existing conditional outputs remains a direction for future work, particularly for conditions excluded from both backbone pretraining and support discovery. Our experiments use classically simulated 16- and 20-qubit systems; the performance and cost of support growth and conditional adaptation at larger scales remain to be evaluated.

7 Conclusion

We introduced ConQuER, a modular post-training interface that enables conditional generation from frozen IQP backbones through composable phase control. Experiments on Binary Blobs, MNIST-3/5, and Shapes20 characterize how support allocation and phase sharing trade conditional distribution quality against controller parameter count. In the fixed-support Blobs extension study, 64 new parameters achieve mean conditional MMD^2 comparable to independent 784-parameter adaptation, with 91.8% fewer new parameters, while existing conditions retain their effective phases. Together, these results support reusing pretrained IQP generators through a conditional interface whose capacity can be allocated, shared, and extended without modifying the stored backbone.

AI USE STATEMENT

We used GPT-6 to assist with background literature review, grammar and sentence-level editing, and the organization of raw experimental results into tables. We reviewed and verified all AI-assisted text and tables and take full responsibility for the final content.

ETHICS STATEMENT

This work studies quantum generative modeling on synthetic Binary Blobs and Shapes20 data and MNIST handwritten digits. It does not involve recruiting human participants, collecting new personal data, or deployment in high-risk applications. We have not identified significant ethical risks specific to this experimental setting. We adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A describe the controller parameterizations and IQP composition. Section 4.1 and Appendix B document data preparation, the computational environment, evaluation metrics, and aggregation across runs. Appendices C–E provide support-construction procedures, optimization schedules, phase-sharing configurations, and new-condition adaptation protocols, including validation-based model selection.

REFERENCES

Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum information*, 5(1):45, 2019.

Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. AkashNarayanan, Ali Asadi, Juan Miguel Arrazola, Utkarsh Azad, Sam Banning, Carsten Blank, Thomas R Bromley, Benjamin A. Cordier, Jack Ceroni, Alain Delgado, Olivia Di Matteo, Amintor Dusko, Tanya Garg, Diego Guala, Anthony Hayes, Ryan Hill, Aroosa Ijaz, Theodor Isacsson, David Ittah, Soran Jahangiri, Prateek Jain, Edward Jiang, Ankit Khandelwal, Korbinian Kottmann, Robert A. Lang, Christina Lee, Thomas Loke, Angus Lowe, Keri McKiernan, Johannes Jakob Meyer, J. A. Montañez-Barrera, Romain Moyard, Zeyue Niu, Lee James O'Riordan, Steven Oud, Ashish Panigrahi, Chae-Yeun Park, Daniel Polatajko, Nicolás Quesada, Chase Roberts, Nahum Sá, Isidor Schoch, Borun Shi, Shuli Shu, Sukin Sim, Arshpreet Singh, Ingrid Strandberg, Jay Soni, Antal Száva, Slimane Thabet, Rodrigo A. Vargas-Hernández, Trevor Vincent, Nicola Vitucci, Maurice Weber, David Wierichs, Roeland Wiersema, Moritz Willmann, Vincent Wong, Shaoming Zhang, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2022. URL https://arxiv.org/abs/1811.04968.

Joseph Bowles, Shahnawaz Ahmed, and Maria Schuld. Better than classical? the subtle art of benchmarking quantum machine learning models. *arXiv preprint arXiv:2403.07059*, 2024.

Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2126):459–472, 2011.

Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum*, 1:8, 2017.

Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi. The born supremacy: quantum advantage and training of an ising born machine. *npj Quantum Information*, 6(1), 2020. ISSN 2056-6387. doi: 10.1038/s41534-020-00288-9. URL http://dx.doi.org/10.1038/s41534-020-00288-9.

Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1), 2018. ISSN 2469-9934. doi: 10.1103/physreva.98.012324. URL http://dx.doi.org/10.1103/PhysRevA.98.012324.

X. Gao, Z.-Y. Zhang, and L.-M. Duan. A quantum machine learning algorithm based on generative models. *Science Advances*, 4(12):eaat9004, 2018. doi: 10.1126/sciadv.aat9004. URL https://www.science.org/doi/abs/10.1126/sciadv.aat9004.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

Harper R. Grimsley, Sophia E. Economou, Edwin Barnes, and Nicholas J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10(1), 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-10988-2. URL http://dx.doi.org/10.1038/s41467-019-10988-2.

Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4884–4896. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.378. URL https://aclanthology.org/2021.acl-long.378/.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Oriel Kiss, Michele Grossi, Enrique Kajomovitz, and Sofia Vallecorsa. Conditional born machine for monte carlo event generation. *Physical Review A*, 106(2), August 2022. ISSN 2469-9934. doi: 10.1103/physreva.106.022612. URL http://dx.doi.org/10.1103/PhysRevA.106.022612.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks, 2015. URL https://arxiv.org/abs/1502.02761.

Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, 98(6):062324, 2018.

Wenjie Liu, Ying Zhang, Zhiliang Deng, Jiaojiao Zhao, and Lian Tong. A hybrid quantum-classical conditional generative adversarial network algorithm for human-centered paradigm in cloud. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):37, 2021.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights, 2018. URL https://arxiv.org/abs/1801.06519.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.08453.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning, 2021. URL https://arxiv.org/abs/2005.00247.

Erik Recio-Armengol and Joseph Bowles. Iqpopt: Fast optimization of instantaneous quantum polynomial circuits in jax. *arXiv preprint arXiv:2501.04776*, 2025.

Erik Recio-Armengol, Shahnawaz Ahmed, and Joseph Bowles. Train on classical, deploy on quantum: scaling generative quantum machine learning to a thousand qubits, 2026. URL https://arxiv.org/abs/2503.02934.

Manuel S. Rudolph, Sacha Lerch, Supanut Thanasilp, Oriel Kiss, Oxana Shaya, Sofia Vallecorsa, Michele Grossi, and Zoë Holmes. Trainability barriers and opportunities in quantum generative modeling. *npj Quantum Information*, 10(1), November 2024. ISSN 2056-6387. doi: 10.1038/s41534-024-00902-0. URL http://dx.doi.org/10.1038/s41534-024-00902-0.

Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2105):1413–1439, 2009.

Ho Lun Tang, V.O. Shkolnikov, George S. Barron, Harper R. Grimsley, Nicholas J. Mayhall, Edwin Barnes, and Sophia E. Economou. Qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum*, 2(2), April 2021. ISSN 2691-3399. doi: 10.1103/prxquantum.2.020310. URL http://dx.doi.org/10.1103/PRXQuantum.2.020310.

Qing-Wei Zeng, Hong-Ying Ge, Chen Gong, and Nan-Run Zhou. Conditional quantum circuit born machine based on a hybrid quantum–classical framework. *Physica A: Statistical Mechanics and its Applications*, 618:128693, 2023.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.05543.

D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, L. Egan, O. Perdomo, and C. Monroe. Training of quantum circuits on a hybrid quantum computer. *Science Advances*, 5(10):eaaw9918, 2019. doi: 10.1126/sciadv.aaw9918. URL https://www.science.org/doi/abs/10.1126/sciadv.aaw9918.

A Supplementary Method Details

A.1 Composition on the union of supports

To expand Eq. (5), write $\mathsf{H} = H^{\otimes n}$ and $D_{\mathcal{G}}(\theta) = \prod_{S\in\mathcal{G}} \exp(-i\theta_S Z_S)$. Extend both phase vectors by zeros to $\mathcal{G}_{\text{all}} = \mathcal{G}_{\text{base}} \cup \mathcal{M}$, writing $\bar{\phi}_m = \text{Scatter}_{\mathcal{M}}(\phi_m)$. Then

$$\begin{aligned} U_{\text{base}}U_{\text{ctrl}}(m) &= \mathsf{H}D_{\mathcal{G}_{\text{base}}}(\theta_{\text{base}})\mathsf{H}\mathsf{H}D_{\mathcal{M}}(\phi_m)\mathsf{H} \\ &= \mathsf{H}\left[\prod_{S\in\mathcal{G}_{\text{all}}} \exp\big(-i[\bar{\theta}_{\text{base},S} + \bar{\phi}_{m,S}]Z_S\big)\right]\mathsf{H}. \end{aligned} \tag{12}$$

Since $\mathsf{H}^2 = I$ and the diagonal gates commute, phases add on common supports. Zero extension includes supports belonging to only one module. Compilation merges gates on overlapping supports and adds any controller supports absent from the backbone.

A.2 Parameter counts and applied output scale

The phase maps in Section 3 store CK values for independent offsets, $d(C + K)$ for the factorized map, and $(K - q) + Cq$ for shared/private control. Table 3 lists the experimental configurations.

For MNIST, an output multiplier α is applied to the controller shift. With an unscaled decoder $\widetilde{W}$, the applied map is $f_\psi(m) = \alpha\widetilde{W}^\top e_m$; setting $W = \alpha\widetilde{W}$ gives Eq. (8). The three training stages use 140, 120, and 100 steps, learning rates 0.002, 0.001, and 0.0007, and output multipliers 2.0, 2.0, and 2.4, respectively. Checkpoint selection evaluates the initial state and recorded training iterates at the fixed generation scale $\alpha = 2.4$, selecting the lowest mean conditional validation MMD^2 and retaining the earliest candidate in a tie. Final evaluation uses the same scale.

Table 3: **Controller parameter counts.** Counts include the independently stored controller values and exclude the frozen backbone, support metadata, and optimizer state. Shared/private configurations retain all K control supports.

Configuration	C	K	Count	P_{ctrl}
Blobs, independent	8	784	8×784	6,272
MNIST, factorized ($d = 10$)	2	136	$10(2 + 136)$	1,380
Shapes20, independent	4	784	4×784	3,136
Blobs, shared/private ($q = 64$)	8	784	$720 + 8 \times 64$	1,232
Shapes20, shared/private ($q = 256$)	4	784	$528 + 4 \times 256$	1,552

B Experimental Details and Metrics

B.1 Data preparation and model references

Binary Blobs. The eight-condition task uses 16-bit samples. A single 15,000-sample realization with generation seed 66 supplies an initial 5,000-sample training pool and a 10,000-sample test set. A class-stratified split of the training pool with seed 20260909 assigns $\lfloor 0.2N_m \rfloor$ examples per condition to validation, where N_m is the condition's pool size. This gives 4,002 training and 998 validation examples. All three backbones and their controllers use these splits. Backbone preparation pools the training samples; adaptation groups them by condition.

Three pooled-MMD checkpoints are expanded to all 65,535 nonempty supports and refined by Hellinger amplitude projection. Each iteration replaces the computational-basis amplitude magnitudes by the square root of the pooled empirical training distribution, retains their phases, and projects the Walsh–Hadamard spectrum back to unit modulus. For backbones b0/b1/b2, Gaussian perturbations of the initial diagonal phases use standard deviations 0.1/0.3/0.1 and RNG seeds 200000/200001/200002. The projection runs for 5,000 iterations, with validation every 50 iterations. Among checkpoints passing the validation MMD, TV, and mode-mass checks, selection minimizes pooled validation MMD^2, yielding iterations 650/1050/1300.

MNIST-3/5. We retain digits 3 and 5 from the official partitions, resize to 4×4 using non-antialiased bilinear interpolation (`align_corners=False`), divide intensities by 255, and binarize at ≥ 0.5. A class-stratified split with seed 20260728 assigns $\mathrm{round}(0.2N_m)$ examples per digit from the official training partition to validation. The resulting splits contain 9,242 training examples (4,905 threes and 4,337 fives), 2,310 validation examples (1,226 threes and 1,084 fives), and all 1,902 official test examples (1,010 threes and 892 fives). The unconditional backbone is trained on a balanced pool of 4,337 examples per digit, subsampling digit 3 without replacement using seed 20260728. Its checkpoint is selected on a balanced validation pool of 1,084 examples per digit. Controller training and checkpoint selection use the full training and validation splits, respectively. The separate-generator reference trains a complete 14,892-parameter IQP for each digit, giving 29,784 parameters across the two models.

Shapes20. Each H/I/T/X class contains nine variants formed by translations and full-row occlusion, with 6% bit noise. The split contains 1,500 training, 400 validation, and 1,500 test samples per class (6,000/1,600/6,000 in total). Backbone pretraining fits the pooled empirical training distribution using all 60,459 supports of orders one through six. An initial phase vector is drawn from $\mathcal{N}(0, 0.0001^2)$. One-qubit entries are then set to $\arcsin \sqrt{\mathrm{clip}(\mu_i, 10^{-4}, 1 - 10^{-4})}$ plus $\mathcal{N}(0, 0.01^2)$ noise; two-qubit entries are set to $0.2\ \mathrm{Cov}(x_i, x_j)$ plus $\mathcal{N}(0, 0.001^2)$ noise, using training means and covariances. Pretraining minimizes pooled MMD^2 with the fixed data-only U correction described in Appendix B.2, using Adam (Kingma & Ba, 2015) ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), learning rate 0.002, and 720 steps. Learning-rate multipliers change from 1 to 0.5 and 0.35 at steps 140 and 260. Backbones b0/b1/b2 use initialization seeds 0/1/2; minimum pooled validation MMD^2 selects checkpoints at steps 720/720/540, respectively.

The main IQP comparison uses backbone b0. ConQuER (fixed supports) fits four independent offset rows on the fixed 196-support dictionary described in Appendix C.1; the main configuration uses the grown 784-support dictionary.

Full IQP fine-tuning initializes one complete phase vector per class from the same pooled-data backbone and optimizes all coordinates. Each vector contains all supports of orders one through six, giving $\sum_{r=1}^{6}\binom{20}{r} = 60{,}459$ parameters per class and 241,836 across the four classes. The conditional AR reference uses raster-order logistic conditionals,

$$p_{\mathrm{AR}}(x \mid m) = \prod_{i=1}^{20} \mathrm{Bernoulli}\left(x_i; \mathrm{sigmoid}\left(b_i + \sum_{j<i} w_{ij}x_j + u_{i,m}\right)\right).$$

It contains 190 previous-bit weights, 80 condition coefficients, and 20 intercepts, totaling 290 parameters. Training minimizes conditional negative log-likelihood with ℓ_2 regularization. The inverse regularization parameter is selected from $\{0.01, 0.1, 1, 10\}$ by mean conditional validation MMD^2.

B.2 MMD EVALUATION

The main evaluations use the Gaussian-kernel mixture $k = \frac{1}{2}(k_{0.6} + k_{1.3})$, where $k_\sigma(x, y) = \exp[-d_H(x, y)/(2\sigma^2)]$, and the two estimators specified in Section 4.1.

Blobs and Shapes20: model probabilities and an empirical target. Let p_m be the complete model probability vector and $\widehat{q}_m$ the empirical test distribution for condition m. Writing $\mathbf{K}_{xy} = k(x, y)$, the V-form metric is

$$\widehat{\mathrm{MMD}}^2_{V,m} = (p_m - \widehat{q}_m)^\top \mathbf{K}(p_m - \widehat{q}_m). \tag{13}$$

Model terms use all 2^{16} probabilities on Blobs and 2^{20} probabilities on Shapes20. The empirical data–data term includes self pairs. Walsh–Hadamard computations evaluate the expression without storing a dense kernel matrix. The Shapes20 test target is the empirical distribution of 1,500 examples per class. Its exact-PMF training loss additionally includes a fixed data-only U-target correction, which leaves parameter gradients unchanged; test scores use the V-form expression above.

MNIST: two-sample U statistic. For each digit, let $X_1, \ldots, X_a$ be generated samples and $Y_1, \ldots, Y_b$ test samples, with $a = b = 512$ for direct generation. Under fixed evaluation seeds, the metric is

$$\begin{aligned}\widehat{\mathrm{MMD}}^2_{U,m} = &\frac{1}{a(a-1)}\sum_{i\neq j} k(X_i, X_j) - \frac{2}{ab}\sum_{i=1}^{a}\sum_{j=1}^{b} k(X_i, Y_j) \\ &+ \frac{1}{b(b-1)}\sum_{i\neq j} k(Y_i, Y_j).\end{aligned} \tag{14}$$

Self pairs are omitted within each sample set.

B.3 CONTROL DIAGNOSTICS AND EMPIRICAL TOTAL VARIATION

Mode assignment and class consistency. For Blobs, $a(x)$ assigns a bitstring to a nearest prototype in Hamming distance, resolving ties by the smallest mode index. The assignment matrix and mean requested-mode hit in Fig. 2(a) are

$$P(j \mid m) = \sum_x p_m(x)\, \mathbb{1}\{a(x) = j\}, \qquad \text{Hit} = \frac{1}{C}\sum_{m=1}^{C} P(m \mid m). \tag{15}$$

For a request-agnostic generator, every row is identical and the diagonal average is $1/C$, giving 12.50% for eight modes.

The main MNIST consistency metric uses a fixed RBF-SVM classifier $h(x) \in \{3, 5\}$ in place of $a(x)$ in the same calculation. The classifier is trained only on training data and attains 75.82% balanced accuracy on real test images. Consistency is computed using complete model probabilities, averaging the two requested digits. The request-agnostic reference is 50%.

Shapes20 target hit is the complete model probability mass assigned to the requested class by a logistic-regression classifier fitted on the adaptation training set. The classifier uses `C=1`, `max_iter=1000`, and `random_state=0`. Predictions select the largest class score, with exact ties resolved by the fixed class order H/I/T/X.

Pixel marginals. The activation probability of pixel i under request m is $r_{m,i} = \sum_x p_m(x)x_i$. Figure 2(b) compares these model marginals with the corresponding empirical averages of real test images.

Empirical total variation. For a model distribution p_m and empirical conditional test distribution $\widehat{q}_m$, total variation is $\mathrm{TV}(p_m, \widehat{q}_m) = \frac{1}{2}\sum_x |p_m(x) - \widehat{q}_m(x)|$. The main MNIST comparison in Table 4 uses complete model probabilities over all 2^{16} bitstrings against the full empirical test distribution of each digit.

Table 4: **Additional MNIST distributional results.** Empirical TV is averaged over the two digits within each run. Trained-model entries report mean ± sample standard deviation over five seeds; the frozen reference is one checkpoint.

Method	Empirical TV ↓
Frozen pooled IQP	0.3345
ConQuER	0.2126 ± 0.0026
Two independent IQPs	0.1520 ± 0.0026

B.4 Aggregation across runs

Condition-level metrics are averaged within each run before aggregating repeated runs. Reported standard deviations are sample standard deviations across those run-level averages. Table 5 lists the evaluation groups in Section 4. The sampling comparison in Appendix C.3 reuses the main evaluation groups.

Table 5: **Evaluation groups and aggregation units.**

Experiment	Repetitions and aggregation
Blobs quality and budget	Eight conditions per backbone, then three backbones with controller seed 0; Fig. 2(c) shows between-backbone standard deviation.
Blobs support comparisons	Backbone 0; three controller seeds for growth, one-shot, and backbone removal; five topology seeds with controller seed 0 for random supports.
Blobs joint sharing	Eight conditions per run, then three backbones with controller seed 0; independent and shared models are retrained on fixed dictionaries.
MNIST trained models	Two digits per run, then five seeds; one frozen reference checkpoint.
Shapes20 main quality	Backbone b0; three controller runs; one run each for full fine-tuning and AR.
Shapes20 dictionary reuse	Backbones b0/b1/b2 and their respective growth-derived dictionaries.
Shapes20 phase sharing	Backbone b0; controller initialization seeds 1 and 2 on fixed supports.
Blobs interface extension	Eight holdouts on each of three backbones; one initialization per configuration, yielding 24 paired cases.

C Support Construction and Budget Studies

C.1 Dictionary construction and optimization budgets

The MNIST dictionary contains every one- and two-qubit support, giving $16 + \binom{16}{2} = 136$ coordinates. Blobs considers all 65,535 nonempty supports on 16 qubits; Shapes20 considers the 60,459 supports of orders one through six on 20 qubits. Each growth configuration uses a common dictionary across conditions.

The main growth path is $256 \to 512 \to 784$, with 120 optimization steps per stage. Starting from an empty dictionary, candidates are first scored at the frozen-backbone phases with zero offsets. At steps 120 and 240, scoring uses the current training parameters. For the condition-specific phase gradients

$g_{m,S}$ defined in Section 3.2, the score is $\sqrt{C^{-1}\sum_m g_{m,S}^2}$. Unused supports are ranked in descending order, with ties resolved lexicographically by their qubit tuples. The selected supports are appended in candidate-pool order, preserving existing coordinates. If the maximum absolute remaining gradient for any condition is at most 10^{-12}, scores are recomputed with a fixed-seed temporary $\mathcal{N}(0, 0.001^2)$ perturbation to candidate phase coefficients. The perturbation is used for scoring only.

Initial offsets are generated as $A_0 = 2.4E_0D_0$, where $E_0 \in \mathbb{R}^{C\times 10}$ and $D_0 \in \mathbb{R}^{10\times K_0}$ have independent $\mathcal{N}(0, 0.01^2)$ entries and K_0 is the initial dictionary size. Optimization updates the entries of A_0 directly as an independent phase table; E_0D_0 specifies only the initialization. At later growth stages, new offsets and their Adam moments start at zero; existing offsets, moments, and optimizer and schedule counters are retained.

The Blobs budget study trains separate configurations at $K \in \{196, 392, 784, 1568, 3136\}$, each with three 120-step stages. The two smaller configurations follow $64 \to 128 \to 196$ and $128 \to 256 \to 392$; the two larger configurations follow $512 \to 1024 \to 1568$ and $1024 \to 2048 \to 3136$. The Adam learning-rate multiplier changes from 1 to 0.5 and 0.35 at global steps 140 and 260. Configuration choices use validation, and the main curve uses controller initialization seed 0 on each of the three backbones.

ConQuER (fixed supports) on Shapes20. The fixed dictionary is constructed once from all pooled training samples and reused across controller runs on backbone b0. After permuting the training rows with RNG seed 8100, Hamming k-medoids with eight centers and seed 0 uses ten restarts, retains sample multiplicities, and selects the lowest training Hamming cost. Each center defines a four-qubit core: its active pixels if exactly four are active, or the four pixels with the highest within-cluster means otherwise. Motifs comprise these cores, their three-qubit subsets, and unions of pairs of cores containing at most six qubits. From the order-one-through-six candidate pool, the dictionary selects 20, 120, 32, 16, and 8 supports of orders 1, 2, 3, 4, and 6, respectively, totaling 196. Within each order, motif supports rank first, followed by descending $\sum_{i<j,\, i,j\in S} |\operatorname{Cov}(x_i, x_j)|$, with qubit tuples breaking ties lexicographically. The four independent offset rows are trained for 360 steps at learning rate 0.02.

C.2 Support-comparison references

Table 6 reports the Blobs backbone-0 comparisons at $K = 784$. One-shot selection scores and selects all 784 supports before fitting. Random supports follow the three-stage growth path and match the growth schedule's interaction-order counts at each stage, drawing unused supports uniformly without replacement within each order. For backbone removal, the backbone phases are set to zero and a newly initialized controller is trained along the fixed support schedule discovered by growth seed 0.

All four methods use 360 steps and the initialization rule above; $K_0 = 784$ for one-shot and 256 for the staged methods. Learning rates are selected from $\{0.002, 0.005, 0.01, 0.02\}$ using mean conditional validation MMD^2. The selected rates are 0.02 for growth, 0.01 for one-shot and backbone removal, and 0.01 for random topology seed 1 versus 0.02 for the other four topology seeds. Validation is checked every 20 steps, with ties retaining the earlier checkpoint. Staged methods select among checkpoints at steps 260, 280, . . ., 360; one-shot also includes initialization and all earlier 20-step checkpoints.

Table 6: **Support comparisons on Blobs backbone 0.** Mean ± sample standard deviation after averaging eight conditions within each run. Growth, one-shot, and backbone removal use controller seeds 0/1/2. Random supports use five topology seeds with controller seed 0. All configurations have $K = 784$ and 6,272 trainable offsets.

Method	$\mathrm{MMD}^2 \downarrow$	Mode hit (%) ↑
Growth / ConQuER	0.009776 ± 0.000072	79.16 ± 0.07
One-shot selection	0.013904 ± 0.001056	75.12 ± 1.07
Random supports	0.261071 ± 0.028388	36.02 ± 3.78
Backbone removal	0.013049 ± 0.000513	81.70 ± 0.60

The separate Shapes20 dictionary-reuse comparison starts from the final supports discovered by growth and fits all offsets from the outset. Across three backbones, mean conditional MMD^2 is 0.02633 for fixed-dictionary fitting and 0.02664 for dynamic growth.

C.3 Direct generation and postselection

We compare direct conditional generation, postselection from the frozen backbone, and postselection from the conditional generator. For source $s \in \{\mathrm{base}, \mathrm{ctrl}\}$, write $p_m^{\mathrm{base}} = p_{\theta_{\mathrm{base}}}$ and $p_m^{\mathrm{ctrl}} = p_{\theta_{\mathrm{eff}}(m)}$. Given a binary acceptance rule $A_m(x)$, the accepted distribution is

$$p_m^{s,\mathrm{acc}}(x) = \frac{p_m^s(x)A_m(x)}{Z_m^s}, \qquad Z_m^s = \sum_x p_m^s(x)A_m(x) > 0. \tag{16}$$

For equally weighted requests, rejection sampling takes $\overline{N}_{\mathrm{raw}}^s = C^{-1}\sum_m (Z_m^s)^{-1}$ raw draws per returned sample in expectation. Direct generation returns each raw draw without filtering.

Both sources accept samples predicted as the requested condition by the same fixed classifier: training-fitted logistic regression on Blobs and Shapes20, and the RBF-SVM on MNIST. Control metrics retain the evaluators in Appendix B.3; Blobs nearest-prototype hit is distinct from logistic acceptance. MMD^2 and TV compare returned outputs with the original label-conditioned test distribution.

Evaluation protocol. The comparison reuses the checkpoints and evaluation groups of Table 1: three Blobs backbones with controller seed 0, five MNIST controller seeds with one frozen backbone, and three Shapes20 controller runs on backbone b0. Blobs and Shapes20 use complete source and accepted probability vectors for MMD^2, TV, and control metrics. MNIST direct-generation MMD^2 follows the sample-based protocol in Appendix B.2. For postselection, SVM rejection filtering retains 1,024 accepted samples from a 32,000-draw evaluation pool for sample-based MMD^2 evaluation. MNIST TV uses complete source or normalized accepted probabilities over all 2^{16} bitstrings against the unchanged full empirical conditional test distributions. $\overline{N}_{\mathrm{raw}}$ is computed from acceptance probabilities.

Table 7: **Direct conditional generation and postselection.** $\overline{N}_{\mathrm{raw}}$ is the expected raw draws per returned sample: accepted outputs for postselection and all outputs for direct generation. Entries are means; where shown, $\pm$ denotes sample standard deviation across controller runs. MNIST and Shapes20 postselection scores are 100% by the acceptance rule.

Sampling method	$\mathrm{MMD}^2 \downarrow$	TV$\downarrow$	Control metric (%)$\uparrow$	$\overline{N}_{\mathrm{raw}} \downarrow$
Binary Blobs: three backbones, controller seed 0				
Frozen + postselection	0.00420	0.229	96.23	8.02
ConQuER direct	0.010494	0.354	78.49	1
ConQuER + postselection	0.02160	0.308	98.28	1.28
MNIST-3/5: five controller seeds				
Frozen + postselection	0.01761	0.329	100	2.07
ConQuER direct	0.00227 ± 0.00020	0.213 ± 0.003	70.39 ± 0.25	1
ConQuER + postselection	0.02208 ± 0.00109	0.337 ± 0.005	100	1.43 ± 0.01
Shapes20: one backbone, three controller runs				
Frozen + postselection	0.00729	0.621	100	4.01
ConQuER direct	0.02416	0.794	64.47	1
ConQuER + postselection	0.01546	0.725	100	1.56

Filtering diagnostics. For this diagnostic, all tasks use complete source and accepted probability vectors and empirical test targets in the quadratic form of Eq. (13); MNIST table entries use the sample-based MMD^2 protocols described above. Writing $\delta_m = p_m^{\mathrm{acc}} - p_m$, filtering changes MMD^2 by

$$\Delta\mathrm{MMD}_m^2 = 2(p_m - \widehat{q}_m)^\top \mathbf{K}\delta_m + \delta_m^\top \mathbf{K}\delta_m. \tag{17}$$

Filtering ConQuER increases MMD^2 in 21 of 24 Blobs backbone–condition cases, while reducing TV in all 24. For each backbone's condition-averaged decomposition, the negative cross term is

outweighed by the nonnegative quadratic term. On MNIST, the SVM accepts 79.1% of digit-3 and 72.5% of digit-5 test examples. Applying this filter to the empirical target distributions themselves induces mean MMD^2 of 0.0183. Filtering ConQuER increases both MMD^2 and TV in all ten seed–digit cases. On Shapes20, filtering improves MMD^2 for all four conditions.

On MNIST, a separate logistic-regression evaluator, not used for filtering, gives requested-digit consistency of 73.26% for frozen-backbone postselection and $83.50\% \pm 0.62\%$ for ConQuER with postselection.

D Phase-sharing Configurations

The fixed-dictionary comparisons use the sharing rules of Section 3.3. Blobs uses gradients at zero offsets: all eight conditions with $q = 64$ for joint training, and the seven available conditions within each extension Stage A. Shapes20 selects $q = 256$ private coordinates using the validation-selected growth seed-0 offset table. Both rules break score ties by increasing dictionary column index.

A fresh random table A_0 is generated as in Appendix C.1 with $K_0 = 784$; shared values start at its column means, and private values retain their condition-specific entries. On each Blobs backbone, independent and shared variants are retrained on its full growth-derived dictionary for 360 steps at learning rate 0.02, with all 784 supports available from the first step. Table 8 reports results across three backbones with controller initialization seed 0. The Shapes20 comparison uses backbone b0 and controller initialization seeds 1 and 2 on its fixed dictionary.

Table 8: **Eight-condition joint sharing on Blobs.** Mean $\pm$ sample standard deviation across three backbones with controller seed 0, after averaging conditions within each run. Both variants use the same fixed dictionary for each backbone. Parameter counts exclude the frozen backbone.

Configuration	Parameters	$\text{MMD}^2 \downarrow$	Mode hit (%) $\uparrow$
Independent offsets	6,272	0.014669 ± 0.002447	74.99 ± 2.16
Gradient-selected ($q = 64$)	1,232	0.009838 ± 0.000818	78.51 ± 0.68

E New-condition Adaptation

E.1 Training, selection, and evaluation protocol

For each of three Blobs backbones and eight conditions h, we reuse the pooled-data backbone and its 784-support dictionary, discovered using all eight conditions. At each private budget $q \in \{64, 128, 256\}$, Stage A trains a shared/private controller from scratch on the seven conditions other than h. Stage B freezes this interface and fits a new private row using labeled data from h. Table 9 specifies both stages; Table 2 reports the $q = 64$ results.

Table 9: **Two-stage extension protocol.** Each stage selects its learning rate and checkpoint using validation data from its training conditions. All trained references use the same Adam optimizer and learning-rate schedule.

Setting	Stage A: shared training	Stage B: new request
Data	Seven available conditions	Held-out condition h
Private coordinates	Selected from the seven conditions' gradients at zero offsets	Retained from Stage A
Trainable values	Shared values and seven private rows	New q-dimensional private row
Learning-rate candidates	$\{0.005, 0.01, 0.02\}$	$\{0.005, 0.01, 0.02, 0.05\}$
Optimization steps	360	360
Initialization	Eight-row random table (seed 0), restricted to seven rows	Mean of the seven fitted Stage-A private rows

Stage A is fitted anew for every budget and holdout. Its initializer is $A_0 = 2.4 E_0 D_0 \in \mathbb{R}^{8\times 784}$, using the Gaussian initialization in Appendix C.1 with $C = 8$ and $K_0 = 784$. NumPy's

`default_rng(0)` draws $E_0 \in \mathbb{R}^{8\times 10}$ followed by $D_0 \in \mathbb{R}^{10\times 784}$. The seven available rows are selected before taking shared-column means and retaining private entries. Validation checks include initialization and every 20 steps, retaining the earlier checkpoint in a tie. For $q = 64$, adding the new private row increases the stored controller values from 1,168 to 1,232; all previously fitted values remain fixed.

The study reuses the fixed Blobs training, validation, and test splits in Appendix B.1. This test split had already been accessed during earlier development. In this extension study, condition h's test subset is evaluated after the validation-selected learning rates and checkpoints for both stages have been recorded. Table 2 provides a descriptive comparison across 24 backbone–holdout cases, reporting mean test error and the number of paired improvements over the independent cold start. Each case has a separately trained Stage-A model; cases share three backbones and overlapping seven-condition training sets.

E.2 Adaptation references

Let ϕ_m^A denote the offset row for condition m decoded from the validation-selected Stage-A checkpoint. The main method and shared-mean references use $\bar{\phi}_{-h} = \frac{1}{7}\sum_{m\neq h}\phi_m^A$. This vector equals the common phase on shared coordinates and the mean of the seven private values on private coordinates. All shared-mean references in Table 2 use the corresponding seven-condition Stage-A model with $q = 64$. Table 10 specifies the adaptations.

Table 10: **New-condition adaptation references.** Every row retains the same frozen backbone and 784-support dictionary. Trained variants use 360 new-condition optimization steps.

Reference	Adaptation configuration
Shared, existing private coordinates	Initialize the new private values from the corresponding entries of $\bar{\phi}_{-h}$ and optimize the existing q private coordinates, keeping Stage-A shared values fixed.
Independent cold start	Initialize from row h of the original eight-row table A_0 used for Stage A and optimize all 784 offsets.
Shared-mean warm start	Initialize a full row at $\bar{\phi}_{-h}$ and optimize all 784 entries.
New-condition-selected coordinates	Initialize at $\bar{\phi}_{-h}$, select the q largest absolute new-condition gradients, and optimize those entries while freezing all others at their initial values.
Shared mean, no training	Apply $\bar{\phi}_{-h}$ directly, with no new fitted values.

For new-condition-selected coordinates, gradients use condition h's training data at the effective phases formed by the backbone and $\bar{\phi}_{-h}$, with ties resolved by dictionary column index.