# Kaiwu-PyTorch-Plugin: Bridging Deep Learning and Photonic Quantum Computing for Energy-Based Models

Anonymous authors
Paper under double-blind review

## Abstract

Kaiwu-PyTorch-Plugin (KPP) connects PyTorch energy-based models to Ising solvers while keeping trainable networks and their objectives in PyTorch. Its reusable Boltzmann-machine interface handles the conversion between model states and solver states, and can be composed with larger models in different roles. We study three such compositions: QDiffusion uses an energy model to guide generation, QVAE uses a Boltzmann prior for latent representations, and Boltz-Flow uses a conditional Boltzmann model to supply the initial state of a flow-matching policy. QDiffusion reaches 439/500 on MATH-500 and a GPT-2 generative perplexity of 22.54 on OpenWebText; Boltz-Flow reaches 98.7% mean success on standard LIBERO. Together with molecular generation and single-cell integration results, these experiments show how the same PyTorch-to-Ising interface supports different roles for an energy model.

## 1 Introduction

Energy-based models (EBMs) score configurations rather than directly predicting normalized probabilities. Boltzmann machines (BMs) and restricted Boltzmann machines (RBMs) are examples (Ackley et al., 1985; Hinton, 2002). BM training needs negative states to estimate a model expectation; other applications use energy to select or initialize outputs. Ising solvers, including coherent Ising machines (CIMs) (Inagaki et al., 2016), search for low-energy states, but their outputs need not follow the model distribution.

For a BM with binary state $s$, $p_\theta(s) = \exp[-E_\theta(s)]/Z_\theta$, where $Z_\theta$ is the partition function. Maximum-likelihood learning compares data and model expectations:

$$\nabla_\theta \mathcal{L}_{\mathrm{NLL}} = \mathbb{E}_{p_{\mathrm{data}}}[\nabla_\theta E_\theta(s)] - \mathbb{E}_{p_\theta}[\nabla_\theta E_\theta(s)]. \tag{1}$$

These are the positive and negative phases. Exact model expectations are generally intractable, so training uses finite negative states. Gibbs sampling follows BM conditionals, whereas an optimization backend searches an Ising landscape and may return low-energy states without sampling $p_\theta$. An Ising instance uses spins $\sigma_i \in \{-1, +1\}$; converting between binary and spin states preserves their energy ordering up to a constant.

Connecting a solver to PyTorch requires mapping current BM parameters to an Ising instance, converting returned spins to model variables, and evaluating a differentiable objective in PyTorch. Earlier quantum-assisted BM systems demonstrate parts of this workflow (Pinilla & Wilton, 2024; Winci et al., 2020). A reusable interface must also support BMs within larger models: a latent prior, a conditional scorer, and a source of initial states share solver access but have different objectives.

We present Kaiwu-PyTorch-Plugin (KPP), which exposes BM energy evaluation and an Ising-solver boundary to PyTorch models. We study three compositions: QDiffusion guides a generative backbone with energy scores, QVAE learns a BM latent prior, and Boltz-Flow uses a conditional BM to initialize a flow-matching action policy. QDiffusion guides generation after proposals are formed; Boltz-Flow changes the source state before flow integration.



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.

We evaluate reasoning, text and molecular generation, single-cell integration, and robot control. These applications demonstrate how a common software boundary supports energy guidance, latent priors, and conditional source distributions across distinct PyTorch models.

## 2 Related Work

Classical BM learning estimates the negative phase with a Markov chain (Ackley et al., 1985). Contrastive divergence uses short Gibbs chains initialized near the data, while RBMs simplify alternating conditional updates (Hinton, 2002). This ties the choice of sampler to the learning procedure. With an external Ising solver, the returned states instead depend on the backend and its settings; the model still evaluates their energies and updates its parameters in PyTorch.

Hardware-assisted BM work has used quantum annealers for training (Pinilla & Wilton, 2024). In QVAE, a neural encoder and decoder are paired with a BM latent prior sampled through an annealer (Winci et al., 2020). CIMs offer an optical route to Ising search (Inagaki et al., 2016; Wang et al., 2013). These studies focus on solver-assisted BM learning in specific systems. KPP makes the Ising mapping and returned-state conversion reusable across model compositions and backends. This interface does not imply that different solvers produce the same state distribution or have the same end-to-end cost.

Other generative models show why an energy component must fit more than a standalone BM training loop. Masked diffusion language models such as MDLM generate sequences by iterative denoising (Sahoo et al., 2024); energy-based diffusion language models add sequence-level energy corrections (Xu et al., 2024). GenMol applies discrete diffusion to molecular strings (Lee et al., 2025), while $\pi_{0.5}$ provides a vision-language-action backbone for robot control (Physical Intelligence et al., 2025). In KPP, QDiffusion scores proposals from these task-specific generators, QVAE uses a BM prior, and Boltz-Flow uses conditional BM states to initialize a flow policy. Their objectives and update rules remain with the parent PyTorch models.

## 3 Problem Setting and Design Requirements

In solver-assisted BM training, positive states come from data or an encoder. Current BM parameters define an Ising instance $I_\theta$ sent to backend $\mathcal{S}$. KPP converts returned spins to binary negative states; PyTorch evaluates positive and negative energies and updates $\theta$.

The solver call is outside PyTorch's differentiable computation. Gradients pass through energy evaluations on returned states, not through the search itself. Two solvers accepting the same Ising instance can return different states and induce different updates, so KPP must preserve the spin-to-variable mapping and experiments must identify the backend and its settings.

The same boundary has different roles in composed models. QVAE trains a BM latent prior alongside an encoder and decoder. BM-backed QDiffusion scores proposed outputs using conditional energy and solver-derived hidden states. Boltz-Flow uses solver-returned states to initialize a flow policy rather than as negative states for training.

These roles require PyTorch to retain trainable networks and objectives, KPP to define explicit Ising and returned-state mappings, and higher-level models to contain an energy component without inheriting the BM interface. A compatible SA or CIM backend can then be selected for the BM component without rewriting the task network.

Interface compatibility does not guarantee equivalent output distributions or latency. For SA, the schedule and run count matter; for a remote CIM, communication and waiting time also matter. We therefore report backend settings, retained-state counts, and filtering separately. Appendix Figure 6 shows the CIM hardware.

# 4 Architecture and Model Composition

## 4.1 System Boundary and Model Structure

Figure 1 shows BM classes and task compositions; Figure 2 traces calls for the three evaluated models. PyTorch runs networks, energy evaluations, and updates; KPP maps BM parameters to Ising instances and solver spins back to model states. The diagram also includes DBN and Gaussian–Bernoulli RBM. Boltz-Flow composes a BM with a flow policy outside the class hierarchy. Appendix Figures 3–5 detail these workflows.

## 4.2 Energy Models and the Solver Contract

AbstractBoltzmannMachine is a PyTorch module that evaluates $E_\theta(s)$, maps its parameters to an Ising instance, and obtains candidate states from a supplied sampler. BoltzmannMachine represents general pairwise interactions; RestrictedBoltzmannMachine limits connections to visible-hidden pairs. Both expose the same objective and state-return path. For a minibatch of positive states $s^+$ and returned negative states $s^-$, KPP computes

$$\mathcal{L}_{\mathrm{BM}}(\theta) = \mathrm{mean}[E_\theta(s^+)] - \mathrm{mean}[E_\theta(s^-)].$$

KPP fixes solver-returned states during backpropagation through their energies. The loss gradient approximates the maximum-likelihood gradient in Eq. 1 only if those states represent the model distribution; low-energy solutions need not.

The same calling pattern accepts SA or CIM backends, but their returned states and costs must be measured separately.

## 4.3 Composing Models from the Core

QVAE places a BM prior within an encoder-decoder model. A relaxed latent variable supports reconstruction, while encoder-derived and sampler-returned latent states contribute to learning the prior. The reconstruction and BM objectives are combined at the application level; QVAE itself does not inherit the BM interface. In masked-token QDiffusion, a proposal network predicts completions from a corrupted sequence and an EnergyModel scores them conditionally. Training contrasts target and proposed completions; generation repeats proposal and energy-guided selection. BM-based scoring can use solver-derived hidden states. For VLA, QDiffusion instead guides a flow-matching action policy, so its proposals and update rule differ from masked-token decoding.

Boltz-Flow conditions a BM on observation and instruction, solves for a binary source state, and decodes it into the initial noise of a flow-matching policy. The policy integrates from that source to an action trajectory. The BM can be trained with a frozen policy, followed by policy adaptation. Unlike QDiffusion, this composition changes the source before flow integration rather than ranking proposals during generation.

The QDiffusion lane in Figure 2 depicts its BM-backed masked-token variant; other generation backbones use task-specific proposals and update rules.

# 5 Experimental Setup

We evaluate QDiffusion, QVAE, and Boltz-Flow with their respective backbones, comparisons, and evaluation protocols.

## 5.1 QDiffusion

QDiffusion guides diffusion backbones in reasoning, text, and molecular generation. Its vision-language-action (VLA) experiment guides a flow-matching policy.

Mathematical reasoning. We use Nemotron-Labs-Diffusion-8B (Fu et al., 2026) as the proposal model and evaluate final-answer accuracy on MATH-500, AIME 2024, AIME 2025,

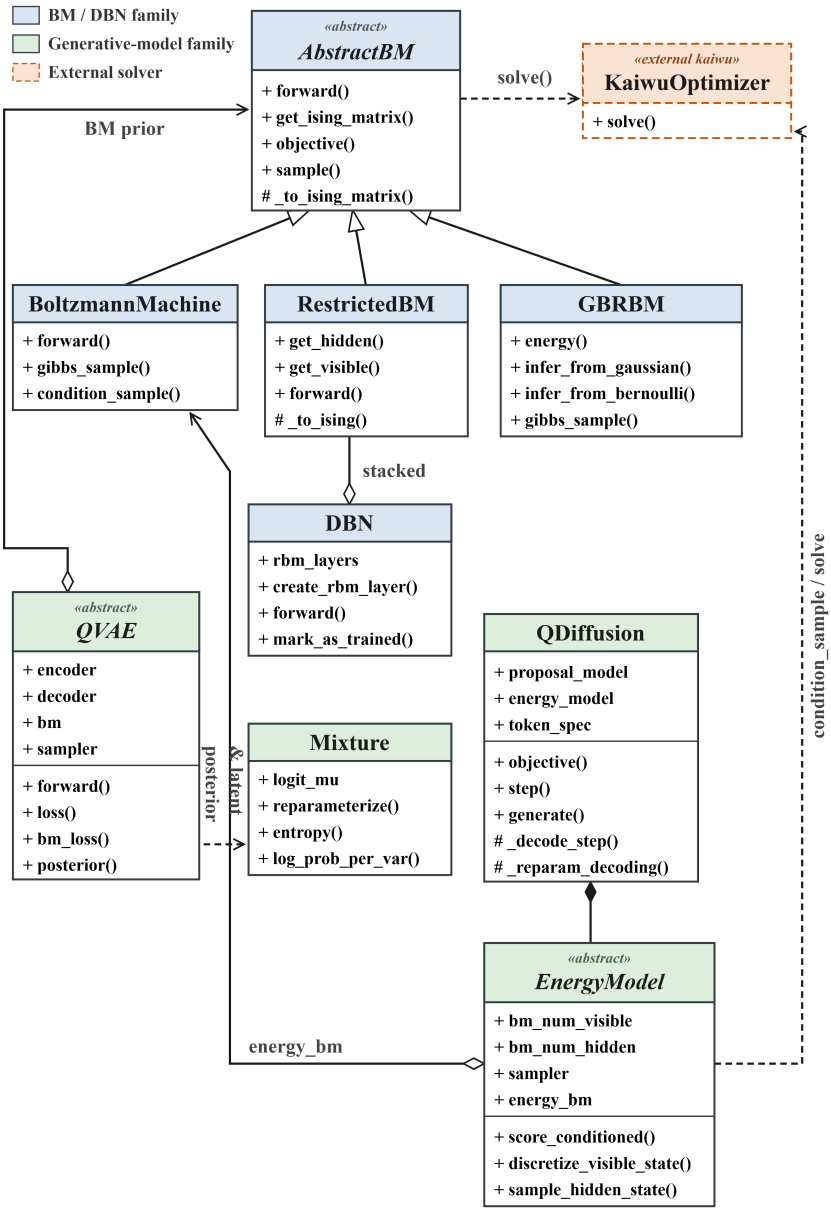


Figure 1: Selected KPP class relationships, with abbreviated class names. BM implementations share the solver boundary; QVAE uses a BM prior, while QDiffusion composes an energy scorer. Boltz-Flow is assembled at the application level.

and GSM8K. Native decoding is the reference configuration. QDiffusion adds a context-dependent Ising energy scorer to select among candidate completions, with the Ising instances evaluated by a CIM backend. To test whether an additional learned scorer alone accounts for the result, we replace the Ising scorer with an MLP scorer. The Native, MLP, and QDiffusion evaluations use the same datasets, decoding settings, and random seeds. We report correct-answer counts with dataset sizes as denominators.

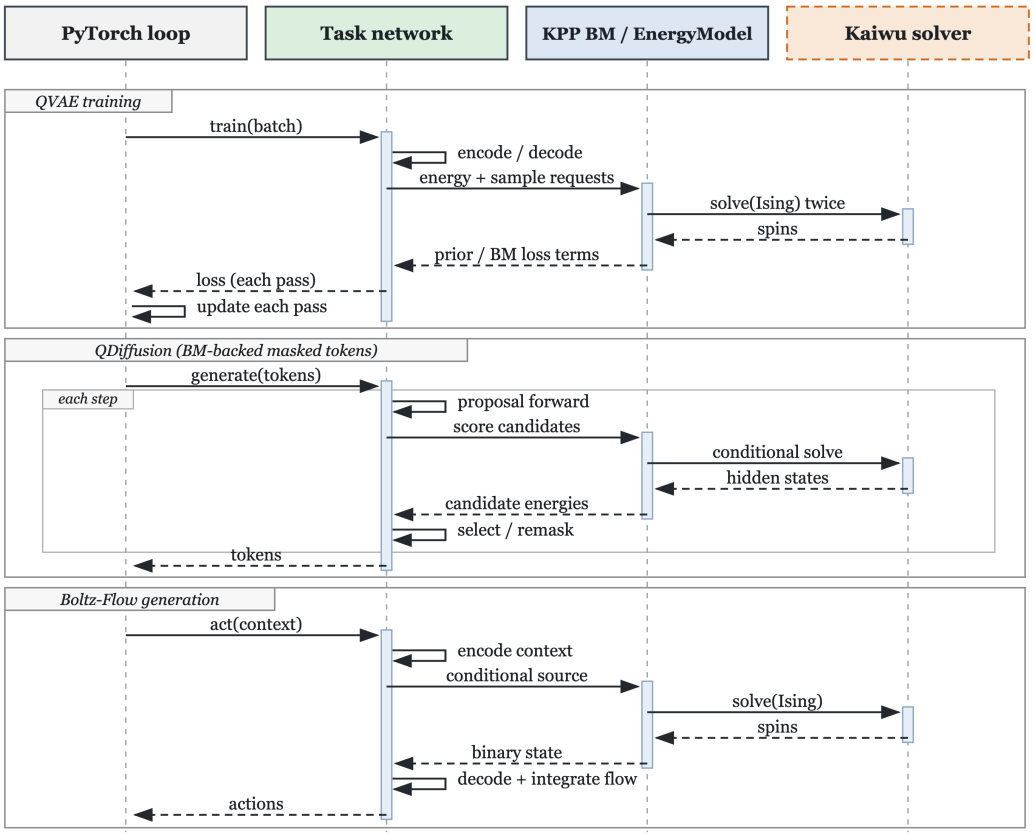


Figure 2: Representative KPP workflows. The QVAE lane groups separate VAE and BM updates, each with a solver call. QDiffusion shows one repeated BM-backed masked-token step. Boltz-Flow decodes returned binary states into noise before PyTorch flow integration.

Open-domain text generation. We use MDLM (Sahoo et al., 2024), trained on OpenWebText, as the diffusion backbone. The QDiffusion configuration adds an energy layer that scores the backbone's candidate completions during generation. We assess generated text using GPT-2 generative perplexity (GenPPL), where lower values indicate better scores under the evaluator. We include MDLM single-candidate baselines at 128 and 1024 steps, the 128-step QDiffusion configuration with two candidates and a 0.1 remasking rate, and the published SEDD result from Xu et al. (2024).

Molecular generation. We additionally study an input-conditioned molecular scorer built on GenMolV2, a pretrained masked diffusion model in the GenMol family (Lee et al., 2025) for SAFE molecular strings. At a guided denoising step, GenMolV2 proposes complete molecular strings. The scorer evaluates each proposal in the context of the current masked string and diffusion time; KPP maps its conditional energy to an Ising instance, obtains a solution from the selected backend, and uses the resulting energy to rank proposals. This is the molecular QDiffusion configuration evaluated here.

The GenMolV2 backbone remains frozen while the energy model learns to rank data completions against generated and perturbed alternatives. The main generation comparison uses simulated annealing and draws two proposals at each guided late step. A base-score selector uses the same two-proposal budget without the learned energy, and an MLP energy provides a second control. The selectors share target-length lists, batching, and paired random-number schedules. We also evaluate the rankers on 900 fixed proposal sets, where every selector sees the same candidates, and run the trained Ising scorer through a direct CIM backend in a separate hardware experiment.

We define molecular Quality as the fraction of requested outputs that are valid, unique by exact SMILES string, and satisfy QED $\geq 0.6$ and a synthetic-accessibility score of at most 4. The main evaluation contains eight streams of 1,000 requested molecules each. Decoding settings are selected on a separate 200-molecule development stream.

VLA action generation. We use the PyTorch OpenPI $\pi_{0.5}$ checkpoint (Physical Intelligence et al., 2025) as the flow-matching backbone. QDiffusion replaces native action generation with BM-guided flow matching; an MLP scoring-head variant provides a control. Standard LIBERO (Liu et al., 2023) comprises Spatial, Object, Goal, and Long, with ten tasks per suite. Each task provides 50 demonstrations. We evaluate 50 episodes per task, giving 2,000 episodes per method, and report the success rate of each suite and their arithmetic mean.

All runs use ten Euler integration steps. We compare the frozen-policy setting with further policy fine-tuning, and evaluate execution intervals of five and ten actions before replanning. LIBERO-Plus (Fei et al., 2025) provides 10,030 held-out test instances across camera, robot, language, lighting, background, sensor, and layout perturbations. We report the mean over these seven factors and a total proxy weighted by the number of instances in each factor. No LIBERO-Plus instance is used for training.

### 5.2 QVAE

QVAE replaces the Gaussian prior with a BM over binary latents. Its encoder maps expression profiles to latent representations; KPP connects the prior to an Ising solver.

We examine single-cell integration on three datasets: Immune, the Human Lung Cell Atlas core dataset (HLCA), and the Human Fetal Lung Cell Atlas (Lung). They contain 33,506, 584,944, and 71,752 cells, respectively. The comparison includes a standard VAE, scVI, LDVAE, and scPoli. QVAE is the QBM-VAE configuration with a KPP BM prior.

We report two aggregate single-cell integration scores. Bio measures how well the integrated representation preserves annotated biological structure, including cell-type separation. Batch measures removal of variation associated with technical batches. Both follow the scIB evaluation, and higher scores are better.

### 5.3 Boltz-Flow

We evaluate Boltz-Flow with the PyTorch OpenPI $\pi_{0.5}$ backbone, LIBERO tasks, LIBERO-Plus test set, and ten-step Euler integration budget used in the VLA study above. A conditional BM scores binary source states given the observation and instruction. Training uses a CIM-based solver; inference uses simulated annealing to reduce latency. The returned state is decoded into the flow policy's initial noise.

The conditional source is trained using offline demonstrations while the base policy is frozen. We also report a configuration with further policy fine-tuning. In the standard LIBERO results, the reported native $\pi_{0.5}$ reference executes five actions before replanning, whereas the Boltz-Flow variants execute ten. All configurations use ten Euler integration steps.

## 6 Results

### 6.1 QDiffusion

Mathematical reasoning. Table 1 reports correct final answers. QDiffusion answers 439 of 500 MATH-500 problems, compared with 429 for Native decoding, a gain of ten problems or 2.0 percentage points. On AIME 2024 and AIME 2025, it answers three additional problems on each set. GSM8K remains at 1207 correct answers out of 1319.

The MLP control does not reproduce the QDiffusion gains under the shared decoding settings. It obtains 362 correct answers on MATH-500 and seven correct answers on each AIME set. These observations indicate that scorer choice matters in this configuration.

Table 1: Correct final answers in mathematical reasoning.

| Dataset | Native | MLP | QDiffusion |
|---|---|---|---|
| MATH-500 | 429/500 | 362/500 | 439/500 |
| AIME 2024 | 8/30 | 7/30 | 11/30 |
| AIME 2025 | 7/30 | 7/30 | 10/30 |
| GSM8K | 1207/1319 | 1206/1319 | 1207/1319 |

Open-domain text generation. Table 2 reports GPT-2 GenPPL on OpenWebText. MDLM scores 43.55 at 1024 steps and 75.22 at 128 steps with $K = 1$; QDiffusion scores 22.72 at 128 steps ($K = 2$, remask 0.1) and 22.54 in its best run.

Table 2: GPT-2 generative perplexity on OpenWebText; lower is better. The SEDD result is from Xu et al. (2024).

| Model and configuration | GPT-2 GenPPL |
|---|---|
| MDLM (1024 steps, $K = 1$) | 43.55 |
| MDLM (128 steps, $K = 1$) | 75.22 |
| SEDD (published) | 33.7 |
| QDiffusion (128 steps, $K = 2$, remask 0.1) | 22.72 |
| QDiffusion (best) | 22.54 |

Molecular generation. Under the matched two-proposal protocol, QDiffusion raises Quality from 0.853 with the diffusion base score to 0.881 (Table 3). The absolute gain of 0.028 corresponds to approximately 225 additional qualifying, exact-string-unique molecules among 8,000 requested outputs. All eight paired stream differences are positive. The paired 95% confidence interval for the mean gain is [0.025, 0.031]. The MLP energy reaches 0.872, between the base score and the Ising scorer.

Table 3: Molecular Quality with two proposals per guided step. Quality is the yield of valid, exact-SMILES-unique molecules meeting the QED and synthetic-accessibility thresholds; higher is better.

| Candidate selector | Quality |
|---|---|
| Diffusion base score | 0.853 |
| MLP energy | 0.872 |
| QDiffusion | 0.881 |

The fixed-candidate evaluation isolates the ranking decision from changes to later generation trajectories. Across 900 proposal sets, exact-sequence recovery increases from 0.0700 without guidance to 0.1922 with the learned energy. In full generation, the Quality gain is associated mainly with more exact-string-unique outputs and a lower mean synthetic-accessibility score. Mean QED decreases slightly, indicating no uniform gain across molecular properties.

In a separate direct-CIM execution study, three generation streams obtain Quality values of 0.914, 0.892, and 0.896, or $0.901 \pm 0.012$ as the mean and sample standard deviation. Matched simulated-annealing streams yield CIM/annealing pairs of 0.914/0.915 and 0.892/0.893. This experiment demonstrates that the trained scorer can execute through KPP's physical CIM backend.

VLA action generation. With the $\pi_{0.5}$ backbone frozen, QDiffusion raises mean standard LIBERO success from 93.7% to 95.2%, versus 93.6% for the MLP control; Long rises from 84.4% to 87.6%. After policy fine-tuning, it raises the five-action mean from 97.4% to 97.9% and the ten-action mean from 97.8% to 98.7%.

On LIBERO-Plus, QDiffusion raises the seven-factor mean from 82.3% to 83.3% with the backbone frozen. Its total proxy rises from 81.4% to 82.4%. The largest factor gains are under sensor noise (+2.3 percentage points) and robot perturbations (+1.7 points). After

Table 4: QDiffusion VLA results on standard LIBERO (%). $r$ is the number of actions executed before replanning; all rows use ten Euler integration steps.

| Setting | Generation | Spatial | Object | Goal | Long | Mean |
|---|---|---|---|---|---|---|
| Frozen, $r = 5$ | Native $\pi_{0.5}$ | 97.6 | 98.6 | 94.0 | 84.4 | 93.7 |
| | MLP guidance | 97.2 | 98.0 | 94.4 | 84.6 | 93.6 |
| | QDiffusion | 98.8 | 99.0 | 95.4 | 87.6 | 95.2 |
| Fine-tuned, $r = 5$ | Native $\pi_{0.5}$ | 98.8 | 99.0 | 98.0 | 93.6 | 97.4 |
| | QDiffusion | 99.2 | 99.2 | 98.0 | 95.2 | 97.9 |
| Fine-tuned, $r = 10$ | Native $\pi_{0.5}$ | 98.0 | 99.2 | 97.8 | 96.0 | 97.8 |
| | QDiffusion | 99.2 | 99.6 | 98.8 | 97.2 | 98.7 |

further fine-tuning, the factor mean rises from 86.1% to 86.8%, while the total proxy rises from 85.0% to 85.8%. Robot perturbations gain the most (+2.0 points).

Table 5: QDiffusion VLA results on LIBERO-Plus (%). Macro averages seven perturbation types. Total proxy is estimated from rounded factor rates and test-instance counts.

| Setting | Generation | Macro | Total proxy |
|---|---|---|---|
| Frozen | Native $\pi_{0.5}$ | 82.3 | 81.4 |
| | QDiffusion | 83.3 | 82.4 |
| Fine-tuned | Native $\pi_{0.5}$ | 86.1 | 85.0 |
| | QDiffusion | 86.8 | 85.8 |

### 6.2 QVAE

QVAE leads Bio and Batch on all three datasets (Table 6); on HLCA, it scores 0.76/0.48 versus scVI's 0.72/0.44. This extends KPP's BM use from guidance to a latent prior.

Table 6: Single-cell integration results for QVAE and reference models. Bio measures biological conservation and Batch measures batch correction; higher is better.

| | Immune | | HLCA | | Lung | |
|---|---|---|---|---|---|---|
| Model | Bio | Batch | Bio | Batch | Bio | Batch |
| VAE | 0.55 | 0.36 | 0.54 | 0.34 | 0.56 | 0.39 |
| scVI | 0.65 | 0.59 | 0.72 | 0.44 | 0.52 | 0.47 |
| LDVAE | 0.55 | 0.58 | 0.65 | 0.43 | 0.53 | 0.47 |
| scPoli | 0.65 | 0.59 | 0.71 | 0.43 | 0.55 | 0.47 |
| QVAE | 0.69 | 0.61 | 0.76 | 0.48 | 0.67 | 0.49 |

### 6.3 Boltz-Flow

On LIBERO (Table 7), frozen Boltz-Flow reaches 98.1% mean success and 96.2% Long, compared with 93.7% and 84.4% for the PyTorch $\pi_{0.5}$ reference. Adaptation raises these to 98.7% and 97.4%. The reference replans after five actions, Boltz-Flow after ten.

On LIBERO-Plus (Table 8), overall success rises from 82.3% to 85.4% with frozen Boltz-Flow and 86.8% after adaptation. The frozen model improves six of seven perturbation types, most under language (+5.3 points), background (+5.1), lighting (+4.9), and layout (+3.9); camera success changes from 69.9% to 69.5%.

Table 7: Standard LIBERO success rates (%). Mean averages four suites. PyTorch $\pi_{0.5}$ replans every five actions and Boltz-Flow every ten. Adapted FT: noise-adapted fine-tuning; †: PyTorch reference; –: unavailable.

| Method | Spatial | Object | Goal | Long | Mean |
|---|---|---|---|---|---|
| Octo | 78.9 | 85.7 | 84.6 | 51.1 | 75.1 |
| Diffusion Policy | 78.3 | 92.5 | 68.3 | 50.5 | 72.4 |
| OpenVLA | 84.7 | 88.4 | 79.2 | 53.7 | 76.5 |
| SpatialVLA | 88.2 | 89.9 | 78.6 | 55.5 | 78.1 |
| WorldVLA | 87.6 | 96.2 | 83.4 | 60.0 | 81.8 |
| CoT-VLA | 87.5 | 91.6 | 87.6 | 69.0 | 83.9 |
| $\pi_0$-FAST | 96.4 | 96.8 | 88.6 | 60.2 | 85.5 |
| SmolVLA-2.25B | 93.0 | 94.0 | 91.0 | 77.0 | 88.8 |
| Qwen-VLA Base | – | – | – | – | 90.8 |
| CogACT | 97.2 | 98.0 | 90.2 | 88.8 | 93.6 |
| $\pi_{0.5}$ (PyTorch)† | 97.6 | 98.6 | 94.0 | 84.4 | 93.7 |
| $\pi_0$ | 96.8 | 98.8 | 95.8 | 85.2 | 94.2 |
| UniVLA | 95.4 | 98.8 | 93.6 | 94.0 | 95.5 |
| Discrete Diffusion VLA | 97.2 | 99.4 | 96.8 | 92.2 | 96.4 |
| StarVLA | 97.8 | 98.6 | 96.2 | 93.8 | 96.6 |
| MemoryVLA | 98.4 | 98.4 | 96.4 | 93.4 | 96.7 |
| GR00T-N1.6 | 97.7 | 98.5 | 97.5 | 94.4 | 97.0 |
| OpenVLA-OFT | 97.6 | 98.4 | 97.9 | 94.5 | 97.1 |
| VITA-VLA | 98.0 | 99.8 | 97.9 | 93.5 | 97.3 |
| $\pi_{0.5}$ + $\text{GridS}_{16}$ | 98.6 | 98.8 | 98.4 | 95.2 | 97.8 |
| Qwen-VLA Instruct | – | – | – | – | 97.9 |
| $\pi_{0.5}$ + Flow SDE | 99.6 | 100.0 | 98.8 | 93.0 | 97.9 |
| $\pi_{0.5}$ + Flow-Noise | 99.6 | 100.0 | 99.6 | 94.0 | 98.3 |
| Boltz-Flow (frozen) | 99.4 | 98.6 | 98.2 | 96.2 | 98.1 |
| Boltz-Flow + adapted FT | 99.2 | 99.8 | 98.4 | 97.4 | 98.7 |

Table 8: LIBERO-Plus success rates (%). Overall averages the seven perturbations; † marks the PyTorch reference.

| Method | Camera | Robot | Language | Light | Background | Noise | Layout | Overall |
|---|---|---|---|---|---|---|---|---|
| OpenVLA | 0.8 | 3.5 | 23.0 | 8.1 | 34.8 | 15.2 | 28.5 | 16.3 |
| WorldVLA | 0.1 | 27.9 | 41.6 | 43.7 | 17.1 | 10.9 | 38.0 | 25.6 |
| NORA | 2.2 | 37.0 | 65.1 | 45.7 | 58.6 | 12.8 | 62.1 | 40.5 |
| UniVLA | 1.8 | 46.2 | 69.6 | 69.0 | 81.0 | 21.2 | 31.9 | 45.8 |
| $\pi_0$ | 13.8 | 6.0 | 58.8 | 85.0 | 81.4 | 79.0 | 68.9 | 56.1 |
| $\pi_0$-FAST | 65.1 | 21.6 | 61.0 | 73.2 | 73.2 | 74.4 | 68.8 | 62.5 |
| OpenVLA-OFT | 56.4 | 31.9 | 79.5 | 88.7 | 93.3 | 75.8 | 74.2 | 71.4 |
| X-VLA | 23.4 | 89.7 | 75.7 | 88.2 | 96.0 | 62.7 | 71.8 | 72.5 |
| MemoryVLA++ | 36.4 | 68.9 | 88.7 | 93.8 | 90.6 | 63.5 | 83.8 | 75.1 |
| VLA-JEPA | 63.3 | 67.1 | 85.4 | 95.6 | 93.6 | 66.3 | 85.1 | 79.5 |
| PokeVLA | 84.7 | 46.1 | 84.8 | 94.6 | 82.6 | 89.8 | 77.2 | 80.0 |
| $\pi_{0.5}$ (PyTorch)† | 69.9 | 73.4 | 80.6 | 91.9 | 92.6 | 85.0 | 82.6 | 82.3 |
| $\pi_{0.5}$ + $\text{GridS}_{32}$ | 76.3 | 67.1 | 76.9 | 95.5 | 92.8 | 90.7 | 82.7 | 83.1 |
| ACoT-VLA | 91.2 | 62.5 | 80.3 | 95.1 | 91.5 | 88.3 | 84.9 | 84.8 |
| Boltz-Flow (frozen) | 69.5 | 76.1 | 85.9 | 96.8 | 97.7 | 85.5 | 86.5 | 85.4 |
| Boltz-Flow + adapted FT | 71.4 | 78.0 | 87.1 | 96.8 | 97.9 | 89.7 | 87.0 | 86.8 |

## 7 Conclusion

KPP provides a reusable PyTorch-to-Ising boundary for energy-based models. QDiffusion, QVAE, and Boltz-Flow use it for generation, latent representation learning, and robot control, respectively, while each task retains its own PyTorch objective.

## AI Use Statement

Generative AI tools, including OpenAI Codex, assisted with experimental design and methodological feedback, implementation, data processing, mathematical derivations, interpretation of results, figure preparation, and the structure, drafting, and editing of this manuscript. The authors checked and verified the AI-assisted code, data processing, deriva-

tions, figures, references, and reported results, reviewed the final manuscript, and take full responsibility for its content.

## References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. Cognitive science, 9(1):147–169, 1985.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Liberoplus: In-depth robustness analysis of vision-language-action models. arXiv preprint arXiv:2510.13626, 2025.

Yonggan Fu, Lexington Whalen, Abhinav Garg, et al. Nemotron-labs-diffusion: A trimode language model unifying autoregressive, diffusion, and self-speculation decoding. Technical report, NVIDIA, 2026. URL https://arxiv.org/abs/2607.05722.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. Neural computation, 14(8):1771–1800, 2002.

Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L McMahon, Takeshi Umeki, Koji Enbutsu, et al. A coherent ising machine for 2000-node optimization problems. Science, 354(6312): 603–606, 2016.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion. arXiv preprint arXiv:2501.06158, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. arXiv preprint arXiv:2306.03310, 2023.

Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. arXiv preprint arXiv:2504.16054, 2025.

Jose P Pinilla and Steven JE Wilton. Quantum-assisted machine learning framework: Training and evaluation of boltzmann machines using quantum annealers. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1, pp. 1688–1694. IEEE, 2024.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. arXiv preprint arXiv:2406.07524, 2024.

Zhe Wang, Alireza Marandi, Kai Wen, Robert L Byer, and Yoshihisa Yamamoto. Coherent ising machine based on degenerate optical parametric oscillators. Physical Review A, 88 (6):063853, 2013.

Walter Winci, Lorenzo Buffoni, Hossein Sadeghi, Amir Khoshaman, Evgeny Andriyash, and Mohammad H Amin. A path towards quantum advantage in training deep generative models with quantum annealers. Machine Learning: Science and Technology, 1(4):045028, 2020.

Minkai Xu, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. Energy-based diffusion language models for text generation. arXiv preprint arXiv:2410.21357, 2024.

## A Detailed Model Workflows

The following sequences expand the BM sampling boundary and the QVAE and QDiffusion paths summarized in Section 4.

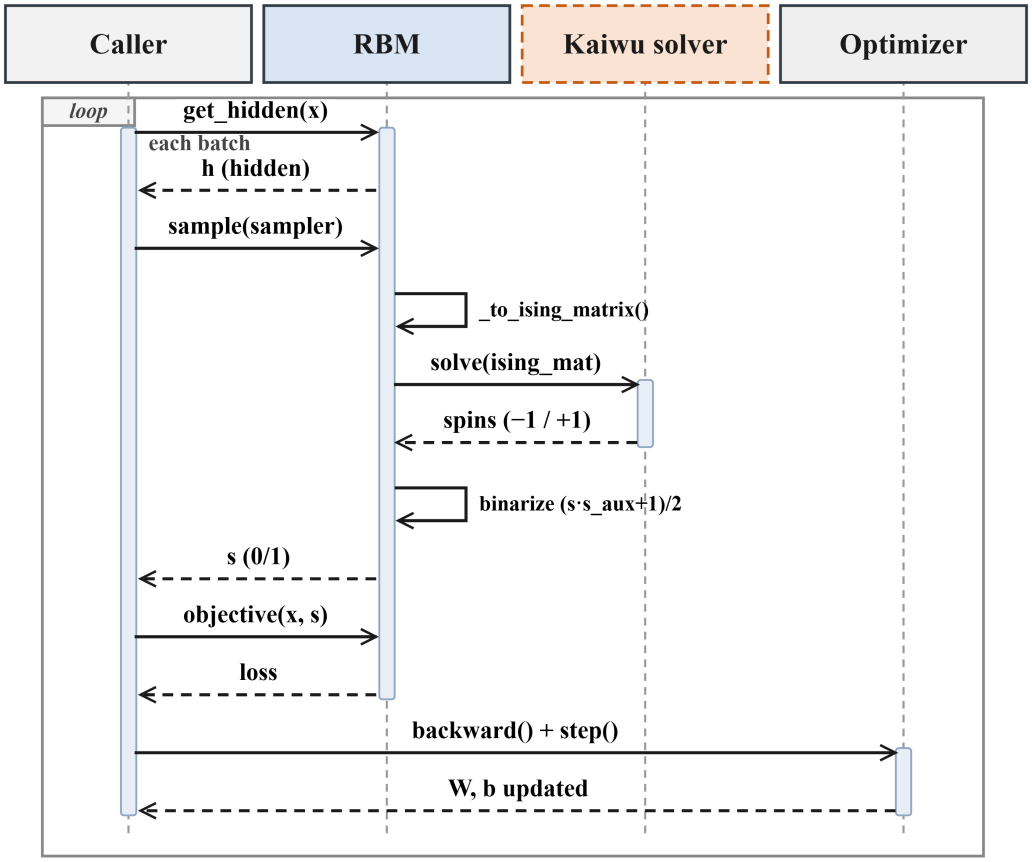


Figure 3: Solver-backed BM sampling and parameter update. Returned spins are converted to binary states before energy evaluation in PyTorch.

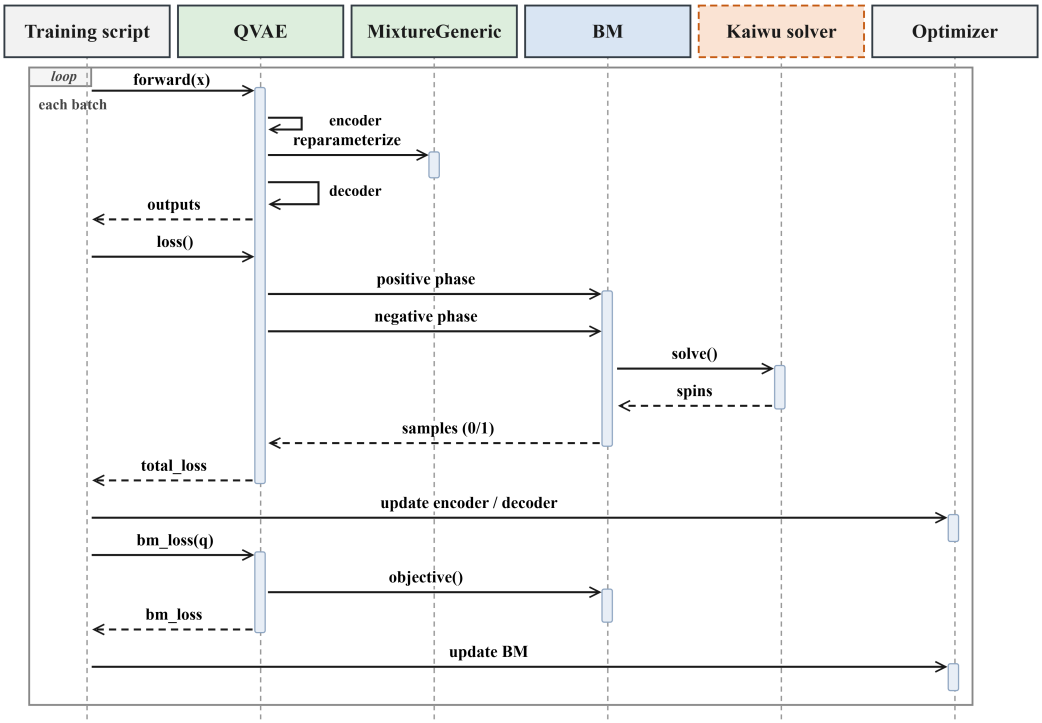


Figure 4: QVAE training sequence with separate reconstruction and BM-prior updates. Both passes draw solver-backed states; the second draw is implicit in the BM-loss call.

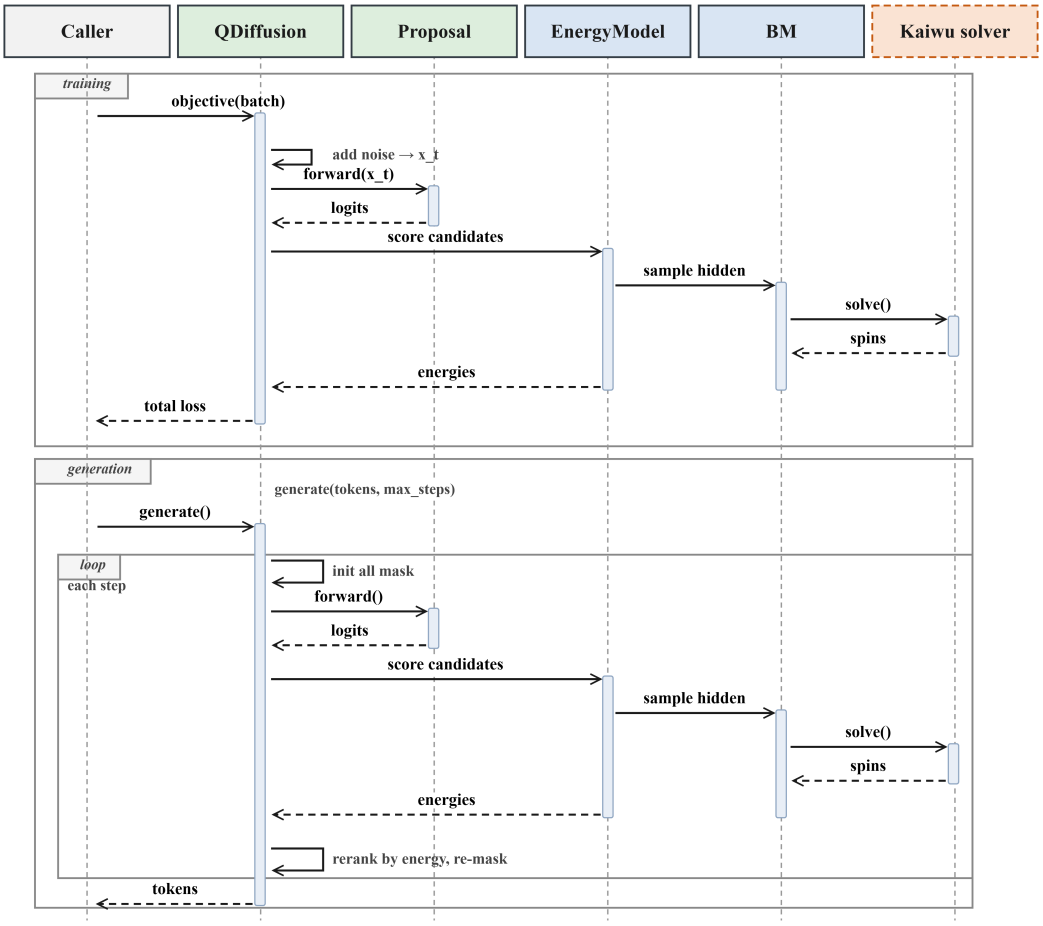


Figure 5: BM-backed QDiffusion sequence for masked-token training and generation. The solver supplies hidden states for conditional candidate scoring.

# B Additional Hardware Details

## B.1 CIM structure

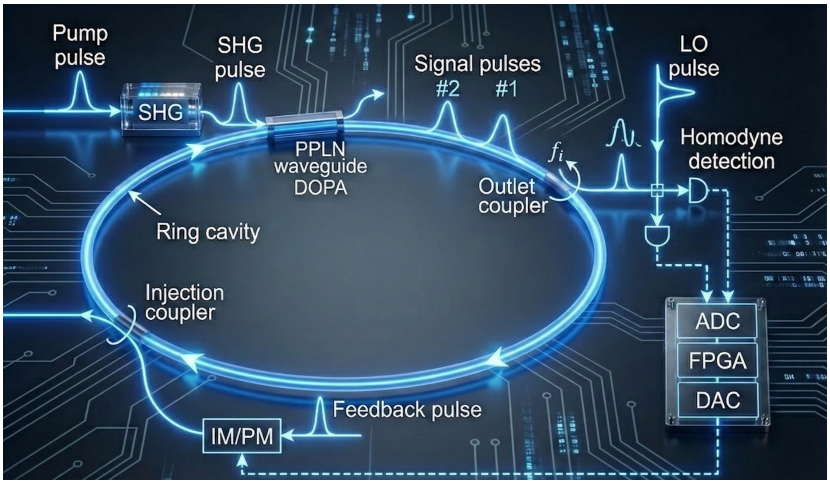


Figure 6: Compact view of the CIM structure.