Training Non-Differentiable Networks via Optimal Transport

Anonymous authors
Paper under double-blind review

Abstract

Neural networks increasingly embed non-differentiable components (spiking neurons, quantized layers, discrete routing, blackbox simulators, etc.) where backpropagation is inapplicable and surrogate gradients introduce bias. We present PolyStep, a gradient-free optimizer that updates parameters using only forward passes. Each step evaluates the loss at structured polytope vertices in a compressed subspace, computes softmax-weighted assignments over the resulting cost matrix, and displaces particles toward low-cost vertices via barycentric projection. This update corresponds to the one-sided limit of a regularized optimal-transport problem, inheriting its geometric structure without Sinkhorn iterations.

PolyStep trains genuinely non-differentiable models where existing gradient-free methods collapse to near-random accuracy. On hard-LIF spiking networks we reach 93.4% test accuracy, outperforming all gradient-free baselines by over 60 pp and closing to within 4.4 pp of a surrogate-gradient Adam ceiling. Across four additional non-differentiable architectures (int8 quantization, argmax attention, staircase activations, hard MoE routing) we lead every gradient-free competitor. On MAX-SAT scaling from 100 to 1M variables, we sustain above 92% clause satisfaction while evolution strategies drop 8–12 pp. On RL policy search we match OpenAI-ES on classical control and retain performance under integer and binary quantization that collapses gradient-based methods. We prove convergence to conservative-stationary points at rate $O(\log T/\sqrt{T})$ on piecewise-smooth losses, upgraded to Clarke-stationary on the headline architectures and extended to the piecewise-constant regime via a hitting-time bound. These rates match the known zeroth-order query-complexity lower bounds that all forward-only methods inherit. Code is available at `https://anonymous.4open.science/r/polystep`.

1 Introduction

Neural networks increasingly incorporate non-differentiable components: hard spike thresholds, quantized activations, blackbox modules, and discrete decision boundaries. Backpropagation requires differentiability; surrogate gradients (Neftci et al., 2019) and the straight-through estimator (STE) (Bengio et al., 2013) introduce approximation bias and require tuning a smooth backward proxy. We seek a gradient-free optimizer that works with any `nn.Module` unmodified.

Prior gradient-free training has a long history. Evolution strategies (CMA-ES (Hansen, 2016), OpenAI-ES (Salimans et al., 2017)) target small-scale supervised learning; zeroth-order methods like DeepZero (Chen et al., 2024) and MeZO (Malladi et al., 2023) scale to deep networks and LLM fine-tuning respectively. All of them assume *differentiable* forward passes where gradients merely become expensive. With genuinely non-differentiable components (hard spikes, integer quantization, discrete routing), small perturbations produce no informative signal and no surrogate is available. The Sinkhorn Step (Le et al., 2023) showed that entropic optimal transport can guide gradient-free updates over polytope geometries, but was demonstrated on low-dimensional motion planning, not on neural network training.

We present PolyStep, a PyTorch optimizer that extends OT-guided polytope optimization to neural network training. The practical update used in all our experiments is a per-particle softmax over polytope-vertex losses, $T^*_{iv} \propto \exp(-C_{iv}/\varepsilon)$, followed by a barycentric step toward low-cost vertices, evaluated entirely via forward passes. This update is the one-sided limit of the regularized OT problem of Le et al. (2023): softmax

recovers OT exactly when the OT target marginal is naturally satisfied (the common subspace regime), and OT strictly extends softmax in the high-particle regime by adding a target-marginal diversity guarantee (an empirical +23 point gap on full-space MNIST; Section 5.10). Standard layers work directly; only attention and recurrent layers require lightweight VmapSafe replacements (Appendix L). To address the dimensionality challenge, PolyStep compresses the parameter space via subspace projections with per-layer decomposition, runs the softmax solver as a single fused kernel per step, and vectorizes forward evaluation via `torch.vmap`.

We report five empirical findings. **(1)** We train models with genuinely non-differentiable forward passes (hard-LIF spikes 93.4%, int8-quantized layers, staircase activations, argmax attention, hard MoE routing) without surrogate gradients or STE; PolyStep beats all gradient-free baselines by $\geq$60 pp on the SNN headline (per-task numbers in Section 5.3). **(2)** On MAX-SAT scaling from 100 to 10^6 variables, PolyStep sustains $> 92\%$ clause satisfaction across four orders of magnitude while CMA-ES and OpenAI-ES drop 8–12 pp; on RL policy search, it matches OpenAI-ES on classical control and retains full performance under INT8/binary quantization that collapses PPO and DQN. **(3)** Being forward-only, PolyStep attains sub-linear memory scaling on recurrent evaluation (29.8$\times$ reduction at T=400) with no activation checkpointing. **(4)** On smooth benchmarks (MNIST 96.0%, ETTh1 0.121 MSE), PolyStep leads all gradient-free methods but Adam retains a clear advantage where exact gradients exist. **(5)** At scale (SST-2, 4.2M params from scratch), every gradient-free method collapses to near-random; we document this intrinsic dimensionality ceiling and tie it to the $\sqrt{d}$ zeroth-order query-complexity lower bounds that any forward-only method inherits.

Our contributions are:

- **Non-differentiable model training**: to our knowledge, the first systematic evaluation of gradient-free optimization on five genuinely non-differentiable architectures, all without surrogate gradients or STE, over 5 seeds. We prove convergence on piecewise-smooth losses (Theorem 4.3), covering each showcase as a corollary. OT serves as the *analytical lens* (the KL-softmax interpolation rate of Theorem 4.1 and the binding-marginal regime witnessed by the +23-pp full-space gap on MNIST); softmax is the practical algorithm we use for every headline number.
- **Beyond supervised learning**: PolyStep trains discrete combinatorial objectives (MAX-SAT at million-variable scale) and RL policies under hard quantization, showing that one optimizer generalizes across supervised, combinatorial, and sequential settings.
- **Forward-only computational model**: each step is a batched forward pass with no gradient tape, no activation storage, and no optimizer state. We demonstrate sub-linear memory on recurrent SNN evaluation (Section 5.4); inference-hardware implications are discussed in Section 6.
- **Open-source PyTorch optimizer**: PolyStep trains standard `nn.Module`s with `torch.vmap`, `torch.compile`, subspace compression, and VmapSafe drop-ins for attention and recurrent layers (Appendix L).

Section 2 reviews related work; Section 3 describes the algorithm; Section 5 reports results; Section 6 discusses limitations and future directions.

2 Background and Related Work

2.1 Gradient-Free Optimization

Zeroth-order (ZO) optimization methods update parameters using only function evaluations, without access to gradients (Nesterov & Spokoiny, 2017). Evolution strategies (ES) sample populations of candidate solutions and update a search distribution based on fitness. CMA-ES (Hansen, 2016) adapts a full covariance matrix, achieving strong performance on low-dimensional problems ($d \lesssim 1000$) but incurring $O(d^2)$ cost. Natural Evolution Strategies (Wierstra et al., 2014) estimate the natural gradient of expected fitness, providing principled updates with lower memory cost. OpenAI-ES (Salimans et al., 2017) simplifies to isotropic perturbations shared via random seeds, scaling to large-population training with low communication cost. Closely related policy-search ES variants include PEPG (Sehnke et al., 2010), which estimates parameter-space gradients from per-sample reward differences with antithetic perturbation pairs, and Augmented Random Search (ARS) (Mania et al., 2018), which uses simple isotropic random search of static linear policies and remains a surprisingly competitive black-box baseline for continuous-control RL. PolyStep's polytope-vertex

probing differs structurally from all of these methods: where ES variants treat each perturbation as an independent gradient sample, PolyStep jointly couples a structured batch of probes through the OT/softmax plan (Section 5.7).

Finite-difference methods estimate gradients from function evaluations. SPSA (Spall, 1992) uses simultaneous random perturbations in two directions, requiring only two evaluations per step regardless of dimension, though with high variance.

Forward-mode methods avoid backward passes entirely: forward gradient descent (Baydin et al., 2022) computes unbiased gradient estimates via forward-mode automatic differentiation, eliminating the backward pass but still requiring differentiable models. MeZO (Malladi et al., 2023) applies in-place ZO-SGD for LLM fine-tuning with inference-level memory. DeepZero (Chen et al., 2024) scales ZO optimization to deep models on CIFAR-10 and ImageNet via coordinate-wise gradient estimation with structured perturbations. PseuZO (Yue et al., 2024) is a memory-efficient ZO method that augments finite-difference perturbations with the gradient of the loss with respect to the model output ($\nabla_o g(o)$); this requires the loss-on-output to be differentiable but allows the model itself to be a black box. Like MeZO and DeepZero, PseuZO is not applicable when the forward pass contains hard thresholds that produce zero output-gradient signal. For high-dimensional ES, LM-MA-ES (Loshchilov, 2017) and dd-CMA-ES (Akimoto & Hansen, 2020) mitigate the $O(d^2)$ covariance cost via low-memory or diagonal-acceleration variants. Recent low-rank ZO methods for LLM fine-tuning, such as GaLore (Zhao et al., 2024) (memory-efficient gradient low-rank projection) and Sparse-MeZO (Liu et al., 2024), reduce the search dimensionality of MeZO-style in-place ZO-SGD by parameter-subset selection or low-rank gradient projection. These methods remain confined to differentiable forward passes and target a different operating regime (pretrained-LLM fine-tuning) than PolyStep's training-from-scratch non-differentiable showcase. Taylor et al. (2016) decompose NN training into gradient-free subproblems via ADMM, but the method requires layer-wise decomposability and differentiable activations within each subproblem. The Forward-Forward algorithm (Hinton, 2022) replaces backpropagation with a local contrastive objective on positive and negative data, avoiding backward passes entirely but requiring architectural modifications. Recent FF-family work (Torres et al., 2025) reports ∼20% test-error reductions and lighter low-capacity-hardware variants, but inherits the same goodness-function architectural constraint. NoProp (Li et al., 2025) eliminates global backpropagation via block-wise denoising objectives, but still uses local gradient computation within each block and requires differentiable layers. Concurrent work on gradient-free quantized training (Cohen et al., 2024) addresses a related setting but uses contribution-tensor heuristics specific to quantized weights rather than a general-purpose optimization framework. Activity perturbation (AP) methods (Dalm et al., 2023) learn by injecting noise into hidden activations and correlating the perturbation with the loss change, a forward-only signal that bypasses the backward pass but still requires differentiable activations to produce informative perturbation responses. Similarly, PEPITA (Dellaferrera & Kreiman, 2022) propagates a perturbation-based error signal through the forward pass, eliminating the need for a symmetric backward path but retaining differentiable layer operations.

These methods share a common assumption: the model's forward pass is differentiable (or at least smooth enough for finite-difference estimation), or (in the case of NoProp) that local gradients within blocks are available. DeepZero's coordinate-wise perturbations require non-zero partial derivatives; MeZO's in-place ZO-SGD assumes the loss varies smoothly with parameters; Forward-Forward and NoProp require custom per-layer objectives or differentiable blocks. When the forward pass contains hard thresholds (sign, round, argmax), these methods produce zero or undefined gradient estimates. Concretely, for a piecewise-constant loss $\mathcal{L}$, the central finite-difference estimate at coordinate i satisfies $\Pr[(\mathcal{L}(\theta + \varepsilon e_i) - \mathcal{L}(\theta - \varepsilon e_i))/(2\varepsilon) = 0] = 1 - O(\varepsilon)$ as $\varepsilon \to 0$: with high probability the 2ε probe interval is smaller than the gap to the nearest threshold, and DeepZero's coordinate-wise estimator returns zero on that coordinate. Methods relying on finite-difference gradient estimation therefore reduce to random search on genuinely piecewise-constant objectives. Our work addresses this complementary setting: models with *genuinely non-differentiable* components where no gradient proxy exists, not even a biased one.

The Sinkhorn Step algorithm (Le et al., 2023) introduced entropic optimal transport as a framework for gradient-free optimization, framing parameter updates as OT problems over polytope geometries. It was demonstrated on low-dimensional motion planning tasks. PolyStep extends this approach to neural network training, addressing the scale challenge via subspace compression and a PyTorch optimizer implementation.

Neural combinatorial optimization. A separate line of work uses neural networks to *learn solution heuristics* for combinatorial problems, including Pointer Networks (Vinyals et al., 2015), NeuroSAT (Selsam et al., 2019), and the broader machine-learning-for-combinatorial-optimization program (Bengio et al., 2021), typically by training a learned solver over a distribution of problem instances. PolyStep's MAX-SAT result (Section 5.5) addresses the orthogonal question of *optimizing the parameters of a single fixed objective* (one MAX-SAT instance treated as a discrete loss landscape) rather than learning a distributional solver. We therefore do not compare to neural-CO methods directly; the comparison ceiling we report (probSAT) is the appropriate domain-specialized SLS reference, and PolyStep is positioned as a general-purpose forward-only optimizer whose million-variable scaling on a non-differentiable landscape is the relevant claim.

2.2 Spiking Neural Networks and Non-Differentiable Models

Spiking neural networks (SNNs) use binary spike events as activations, introducing hard non-differentiability at each spike threshold. Surrogate gradient methods (Neftci et al., 2019) circumvent this by replacing the spike function with a smooth surrogate during the backward pass, enabling gradient flow through temporal unrolling (BPTT). Surrogate gradients achieve high accuracy on neuromorphic benchmarks but require storing the full computational graph across T timesteps: $O(T)$ memory.

Memory-efficient BPTT variants reduce this cost while retaining gradient-based updates. Spatial Learning Through Time (SLTT) (Meng et al., 2023) achieves near-$O(1)$ memory by pruning unimportant temporal backpropagation routes, providing an important contrast to our work: SLTT achieves efficient memory *with* gradient information, while PolyStep achieves sub-linear memory scaling *without* any gradient access, targeting settings where no gradient proxy exists.

Blackbox and quantized networks present related challenges. Binary and ternary weight networks (Hubara et al., 2016) use the straight-through estimator (STE) (Bengio et al., 2013) to approximate gradients through the sign function. Yin et al. (2019) prove that STE's coarse gradient correlates positively with the population gradient under restrictive assumptions (binary ReLU networks, Gaussian input), and Jeong et al. (2025) extend this to finite-sample analysis for two-layer binary networks. STE is therefore understood under restrictive structural assumptions but remains fragile in general: the bias depends on the surrogate slope and the data distribution, and no analogous theory covers hard MoE routing or staircase activations. No general gradient-based solution exists for hard quantization, discrete routing, or non-differentiable decision boundaries. NITRO-D (Pirillo et al., 2025) sidesteps the differentiability barrier in the integer-only training direction by replacing the optimizer and activation primitives with native-integer counterparts; this is complementary to our forward-only approach in that it constructs a differentiable training pipeline operating in the integer domain, whereas PolyStep keeps the original forward pass intact and avoids gradient computation altogether. Zeroth-order methods are a natural fit in these settings, provided the search dimensionality is manageable, yet prior work on ES and ZO methods has focused almost exclusively on differentiable networks. Our experiments specifically target models where the non-differentiability is in the forward pass, not just in the loss landscape. We note that recent memory-efficient zeroth-order optimizers such as MeZO (Malladi et al., 2023) and DeepZero (Chen et al., 2024) are not directly comparable: both estimate gradients via finite-difference perturbation through the model's forward pass, which requires the forward pass itself to be differentiable (or at least Lipschitz continuous) for the gradient estimate to carry meaningful signal. On genuinely non-differentiable architectures (hard LIF spikes, rounding, sign thresholds, argmax routing), the perturbation-based gradient estimate is zero almost everywhere, reducing these methods to random search. PolyStep sidesteps this failure mode entirely by treating the forward pass as a black-box cost oracle.

2.3 Optimal Transport for Optimization

Entropic optimal transport regularizes the classical Kantorovich problem with a KL divergence term, yielding the Sinkhorn algorithm (Cuturi, 2013) as an efficient solver. For a comprehensive treatment of computational OT, see Peyré & Cuturi 2019. The equivalence between scaled-dot-product attention and one-sided entropic OT (Litman, 2025) crystallizes a folklore observation in the OT community (Cuturi, 2013; Peyré & Cuturi, 2019): the row-marginal-only softmax is exactly the closed-form $\lambda \to 0$ endpoint of the KL-penalized unbalanced OT framework of Chizat et al. (2018). The doubly-stochastic generalization, Sinkformers Sander et al.

(2022), replaces softmax with the full Sinkhorn fixed point and provides the canonical earlier formalization of attention as OT, with a Wasserstein-gradient-flow interpretation in the infinite-depth limit.

OT has been applied to generative modeling, domain adaptation, distribution matching, and learning latent permutations via Gumbel-Sinkhorn networks (Mena et al., 2018); our setting is distinct: we use OT over finite polytope vertex sets to select descent directions in parameter space, rather than as a differentiable relaxation of combinatorial structures.

The Sinkhorn Step (Le et al., 2023) established this connection formally, proving that the barycentric projection of the OT plan over polytope probes provides a principled descent direction for smooth objectives. PolyStep inherits this theoretical foundation and adds the engineering required for NN-scale problems: subspace compression with per-layer projections, `torch.vmap` vectorization for batched forward evaluation, and `torch.compile` acceleration of the inner Sinkhorn solver.

3 Algorithm

We now describe the PolyStep algorithm for gradient-free neural network training. Given a model f_θ with parameters $\theta \in \mathbb{R}^d$ and a loss function $\mathcal{L}$, the optimizer uses only forward-pass evaluations $\theta \mapsto \mathcal{L}(f_\theta)$ to update parameters; no gradient oracle is required. This enables training models with non-differentiable components (spiking neurons, quantized layers, black-box modules) where backpropagation is inapplicable or prohibitively expensive.

3.1 Problem Formulation

We seek to minimize $\mathcal{L}(\theta)$ using only function evaluations. PolyStep makes *no assumptions* on $\mathcal{L}$: the loss need not be differentiable, continuous, or even Lipschitz; it need only be evaluable as a scalar. This is strictly more general than zeroth-order gradient methods (which assume smoothness for finite-difference estimates) and STE/surrogate approaches (which require a differentiable proxy). The model parameters $\theta \in \mathbb{R}^d$ are reshaped into a *particle matrix* $\boldsymbol{X} \in \mathbb{R}^{P \times d_p}$, where $P = \lceil d/d_p \rceil$ is the number of particles and $d_p \in \{2, 4, 8\}$ is the particle dimension. Each row x_i of $\boldsymbol{X}$ represents a particle (a contiguous slice of the parameter vector), and the optimization operates on all particles simultaneously.

Given a cost matrix $\boldsymbol{C} \in \mathbb{R}^{P \times V}$ scoring each (particle, polytope vertex) pair, the central optimization primitive is a temperature-controlled soft assignment from particles to vertices. We use this assignment to construct a barycentric step (Eq. 6) that displaces each particle toward low-cost vertices. PolyStep provides two solvers for this assignment, related by a strict generalization:

- **Softmax (default in all reported experiments).** Each particle i assigns weights to vertices independently:
$$T^*_{iv} \;=\; a_i \cdot \mathrm{softmax}(-C_{i:}/\varepsilon)_v\,, \qquad a_i = 1/P, \tag{1}$$
yielding row sums $\sum_v T^*_{iv} = a_i$ and a closed-form, single-pass update. This is the update rule used for every headline number in the paper.
- **Entropic optimal transport (principled generalization).** The softmax rule is the one-sided limit of the regularized OT problem
$$\min_{\boldsymbol{T} \geq 0} \langle \boldsymbol{C}, \boldsymbol{T} \rangle + \varepsilon \cdot D_{\mathrm{KL}}(\boldsymbol{T} \| \boldsymbol{a} \otimes \boldsymbol{b}), \quad \text{s.t. } \boldsymbol{T}\mathbf{1} = \boldsymbol{a},\ \boldsymbol{T}^\top \mathbf{1} = \boldsymbol{b}, \tag{2}$$
where $\boldsymbol{a} \in \mathbb{R}^P$ and $\boldsymbol{b} \in \mathbb{R}^V$ are uniform marginals with $a_i = 1/P$ and $b_v = 1/V$, and $\varepsilon > 0$ is the entropic regularization parameter (Cuturi, 2013; Peyré & Cuturi, 2019). Adding the target marginal $\boldsymbol{T}^\top \mathbf{1} = \boldsymbol{b}$ couples particles via a price signal g_v (the dual potential) that prevents many-particle collapse onto a single low-cost vertex. When the number of particles is small relative to the vertex set (e.g., subspace mode), the target constraint is naturally satisfied, $g_v \approx 0$, and entropic OT reduces *exactly* to the softmax rule (Eq. 1) (Litman, 2025).

The parameter ε decays on a schedule (e.g., cosine from 3.0 to 0.1) and controls both the exploration radius (via $r_s\varepsilon$ and $r_p\varepsilon$) and the soft-assignment sharpness (Le et al., 2023). Theorem 4.3 is proved for the canonical

schedule $\varepsilon_t = \varepsilon_0/\sqrt{t+1}$; the cosine schedule used in our experiments is one of several alternatives ablated in Section 5.10. We use the softmax solver throughout Section 5; the OT generalization is empirically motivated and ablated in Section 5.10 (Tables 10 and 11; see Section 5.10 for the full update-rule and regime-map results), where it outperforms softmax by 23 points on full-space MNIST (P=50,885 particles competing for V=4 vertices).

We define the following notation used throughout the paper:

- P: number of particles ($= \lceil d/d_p \rceil$ in full-space mode)
- d_p: particle dimension (default 2; increasing to 4 enriches the OT signal)
- V: number of polytope vertices ($2d_p$ for the orthoplex)
- K: number of probe points per vertex (default 1)
- ε: entropic regularization / exploration radius scale
- r_s, r_p: step radius and probe radius multipliers
- d_{sub}: effective optimization dimension. In full-space mode, $d_{\text{sub}} = d$ (all parameters); in HybridSubspace, $d_{\text{sub}} = \sum_l (d_{\text{out},l} + d_{\text{in},l})\, r_l$ where $r_l = \min(r, d_{\text{out},l}, d_{\text{in},l})$ is the effective rank per layer (plus bias dimensions included at full size). The number of particles is $P = \lceil d_{\text{sub}}/d_p \rceil$

In full-space mode, the total number of forward-pass evaluations per step is $P \times V \times K$. For the orthoplex ($V = 2d_p$) with K=1, each step evaluates $2d_{\text{sub}}$ probe points; with K=3 (the default in most experiments), the cost triples to $6d_{\text{sub}}$.

3.2 Core Optimization Step

Each optimization step consists of five sub-steps: polytope sampling, probe generation, cost matrix construction, soft-assignment solve, and barycentric projection. We describe each in turn and summarize the full procedure in Algorithm 1 (Figure 1 provides a geometric illustration).

Step 1: Polytope sampling with random rotations. For each particle i, we sample a rotation matrix $R_i \sim \text{Uniform}(\text{SO}(d_p))$ via the Mezzadri QR method (Mezzadri, 2007) (or an exact analytical rotation for $d_p = 2$). Three polytope templates define the vertex set $\mathcal{V}$:

- **Orthoplex** (cross-polytope): $2d_p$ vertices $\{\pm e_j\}_{j=1}^{d_p}$. Default; ablation confirms best accuracy.
- **Simplex**: $d_p + 1$ vertices.
- **Cube** (hypercube): 2^{d_p} vertices. Exponential in d_p; recommended only for $d_p \leq 4$.

The rotated step vertices for particle i are:

$$v_{i,j} = x_i + r_s \varepsilon\, R_i\, v_j, \quad v_j \in \mathcal{V}. \tag{3}$$

Step 2: Probe generation. For each particle–vertex pair (i, v), we generate K probe points along the rotated vertex direction $R_i v_j$, scaled by an independently sampled radius jitter $\eta_t \sim \text{Uniform}[-\eta_{\max}, \eta_{\max}]$ with $\eta_{\max} \in [0, 1)$:

$$p_{i,v,k} = x_i + r_p(1 + \eta_t)\varepsilon\, R_i\, v_j\, \lambda_k, \quad \lambda_k = \frac{k}{K+1},\ k = 1, \ldots, K. \tag{4}$$

The probes sample the loss landscape at intermediate distances along the same directions as the step vertices, but at an independently controlled radius $r_p(1+\eta_t)\varepsilon$ (rather than at fractions of the step vertex distance $r_s\varepsilon$). The radius jitter η_t adds the missing dimension of randomness needed for the joint (R_i, η_t) probe distribution to be absolutely continuous on a positive-measure tube around the $(d_p$–1)-sphere; this transversality property is what the convergence proof of Theorem 4.3 requires (Section 4.2). The convergence theorem formally requires $\eta_{\max} > 0$; in our implementation the jitter defaults to 0 so the reported experiments are bit-for-bit reproducible, and we recommend $\eta_{\max} = 0.05$ for new applications, a value that lies well within the measured probe-radius sensitivity envelope (Section 5.10) and therefore does not move headline numbers materially. When $K = 1$, each probe is placed at half the (jittered) probe radius along the rotated direction: $p = x_i + \frac{1}{2} r_p(1 + \eta_t)\varepsilon\, R_i\, v_j$.

Algorithm 1: PolyStep Optimization Step (softmax solver, default in all reported experiments)

Input: Model f_θ, loss $\mathcal{L}$, temperature ε, radii r_s, r_p, probes K
Output: Updated θ', transport cost
$\boldsymbol{X} \leftarrow$ params_to_particles(θ) ; // $\boldsymbol{X} \in \mathbb{R}^{P \times d_p}$
for *each particle* $i = 1, \ldots, P$ **do**
  Sample $R_i \sim \mathrm{Uniform}(\mathrm{SO}(d_p))$, $\eta_t \sim \mathrm{Uniform}[-\eta_{\max}, \eta_{\max}]$;
  $\{v_{i,1}, \ldots, v_{i,V}\} \leftarrow r_s \varepsilon\, R_i \cdot \mathrm{Polytope}(d_p) + x_i$; `// step vertices`
  for *vertex* $j = 1, \ldots, V$, *probe* $k = 1, \ldots, K$ **do**
    $p_{i,j,k} \leftarrow x_i + r_p(1 + \eta_t)\varepsilon\, R_i\, v_j\, \lambda_k$; `// radius jitter for transversality (Theorem 4.3)`
$C_{iv} \leftarrow \frac{1}{K} \sum_k \mathcal{L}(f_{\theta(p_{i,v,k})})$; `// vectorized via vmap;` $\boldsymbol{C} \in \mathbb{R}^{P \times V}$
$T^*_{iv} \leftarrow (1/P)\,\mathrm{softmax}(-C_{i:}/\varepsilon)_v$; `// single fused kernel; default solver`
$x_i^{\mathrm{new}} \leftarrow \frac{1}{a_i} \sum_v T^*_{iv}\, v_{i,v}$; `// barycentric projection`
$\theta' \leftarrow$ particles_to_params($\boldsymbol{X}^{\mathrm{new}}$);
return θ', $\sum_{iv} C_{iv}\, T^*_{iv}$

Step 3: Cost matrix construction. All probe points are evaluated in parallel via `torch.vmap` (Appendix L). The cost entry for particle i and vertex v averages the probe losses:

$$C_{iv} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}\big(f_{\theta(p_{i,v,k})}\big)\,. \tag{5}$$

The resulting cost matrix $\boldsymbol{C} \in \mathbb{R}^{P \times V}$ has dimensions particle-by-vertex, explicitly *not* $P \times P$. The OT problem transports mass from particles to vertices, not between particles, which makes the Sinkhorn solve $O(P \cdot V)$ rather than $O(P^2)$.

Computational model: batched inference. Each PolyStep step evaluates $P \times V \times K$ candidate parameter configurations via a single batched forward pass, the same computational primitive as batched model inference. Unlike ES methods that perturb parameters and collect scalar returns with independent noise samples, PolyStep's polytope geometry produces a structured batch of candidates that can be evaluated in parallel via `torch.vmap` or explicit `torch.bmm` (for MLP-only models). The per-step cost is therefore dominated by a single batched inference call, which modern GPU architectures and the emerging class of inference-optimized accelerators are specifically designed to maximize (Section 6).

Step 4: Soft-assignment solve. Default (softmax). The transport plan is computed in a single pass via Eq. 1: $T^*_{iv} = (1/P) \cdot \mathrm{softmax}(-C_{i:}/\varepsilon)_v$. This is one fused softmax kernel per step (no iterations, no dual potentials), used in all reported experiments. **OT generalization.** The softmax rule is the $\lambda \to 0$ endpoint of a solver continuum: softmax $\to$ KL-softmax (Theorem 4.1) $\to$ full entropic OT (Eq. 2). Setting the Sinkhorn dual potential $g \equiv 0$ recovers softmax exactly; nonzero g enforces the target marginal and provides a diversity guarantee that matters when many particles compete for few vertices (Table 11, +23 points on full-space MNIST). The full log-domain Sinkhorn update equations, warm-starting strategy, overrelaxation, and low-rank factorization details are provided in Appendix K.

Step 5: Barycentric projection. Each particle moves toward the low-cost vertices, weighted by the optimal transport plan:

$$x_i^{(t+1)} = \frac{1}{a_i} \sum_{v=1}^{V} T^*_{iv} \cdot v_{i,v}. \tag{6}$$

The updated particle matrix $\boldsymbol{X}^{\mathrm{new}}$ is then mapped back to the model's parameter dictionary via the inverse of the flatten operation.

Remark 3.1 (Generalization to entropic optimal transport). Algorithm 1 produces every headline number in this paper. To use the principled generalization, replace the softmax line with the entropic-OT solve $\boldsymbol{T}^* \leftarrow \mathrm{Sinkhorn}(\boldsymbol{C}, \varepsilon, \omega, f_{\mathrm{prev}}, g_{\mathrm{prev}})$ of Eq. 2, with overrelaxation $\omega \in [0.5, 1.95]$ and warm-started dual

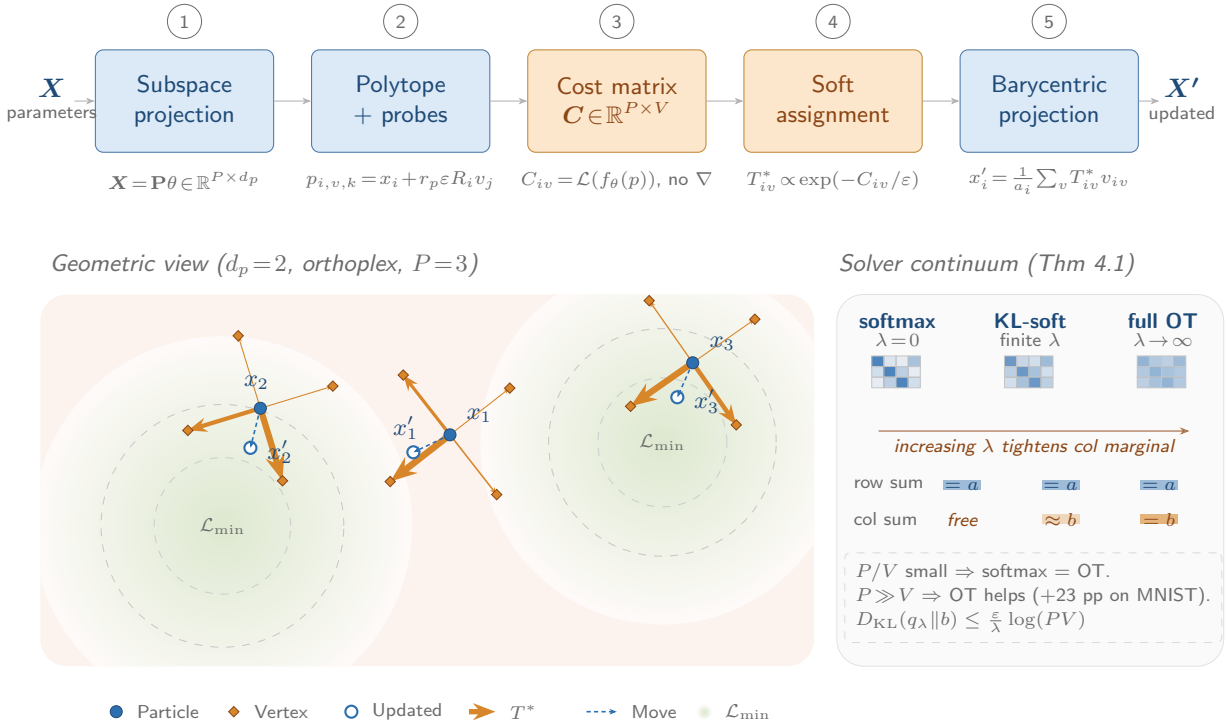

Figure 1: One PolyStep optimization step. **Top:** forward-only pipeline. Parameters θ are compressed into P particles via subspace projection (1), polytope vertices are sampled around each particle under random rotations (2), the cost matrix is evaluated by forward passes only (3), the soft-assignment T^* is computed by either softmax ($\lambda = 0$) or entropic OT ($\lambda \to \infty$) (4), and particles update via barycentric projection (5). **Bottom-left:** geometric view in two dimensions with three particles (orthoplex, $d_p = 2$); arrow width is proportional to T^*_{iv}, and each particle steps toward the lowest-cost vertex inside its polytope, drawn here on a non-convex landscape with two local minima. **Bottom-right:** solver continuum (Theorem 4.1): softmax recovers the row-marginal-only OT plan; increasing λ tightens the column marginal toward $\mathbf{b}$, yielding the full entropic OT plan in the limit. The softmax solver is used for every reported headline number; full OT yields a +23 pp gain in the high-particle regime where the column marginal is binding (Section 5.10).

potentials $(f_{\text{prev}}, g_{\text{prev}})$ from the previous iteration. The entropic-OT path adds the target-marginal column constraint $\boldsymbol{T}^\top \mathbf{1} = \boldsymbol{b}$, which is non-binding in the subspace regime and yields a +23 pp accuracy advantage in the high-particle full-space regime (Section 5.10, Table 11). Full log-domain Sinkhorn equations, warm-starting strategy, and low-rank factorization are in Appendix K.

3.3 Subspace Compression

In full-space mode, the number of particles $P = \lceil d/d_p \rceil$ grows linearly with model size, making the cost matrix $\boldsymbol{C} \in \mathbb{R}^{P \times V}$ and the Sinkhorn solve expensive for large models. Subspace compression decouples P from d by projecting parameters into a fixed-size subspace before running Algorithm 1. Random subspace projections have proven effective for zeroth-order optimization at scale: SubZero (Yu et al., 2024) uses layer-wise low-rank perturbations for LLM fine-tuning, and MeZO (Malladi et al., 2023) operates in the full parameter space with in-place perturbations. PolyStep supports four projection modes, summarized in Table 1.

HybridSubspace (recommended). Each weight layer l with shape $(d_{\text{out}}, d_{\text{in}})$ receives an independent dense projection matrix $P_l \in \mathbb{R}^{d_l \times n_l}$, where $n_l = (d_{\text{out}} + d_{\text{in}})\, r_l$ follows a factored low-rank parametrization with effective rank $r_l = \min(r, d_{\text{out}}, d_{\text{in}})$, and entries are drawn i.i.d. from $\mathcal{N}(0, 1/n_l)$. Bias parameters are included at full dimension (identity projection, no compression). Reconstruction maps the subspace coordinate $z_l \in \mathbb{R}^{n_l}$ back to the full layer: $\delta\theta_l = P_l\, z_l$. The total subspace dimension is $d_{\text{sub}} = \sum_l n_l + \sum_b d_b$

Table 1: Subspace projection modes. d is total parameter count, r is rank, $n_l = (d_{\text{out},l} + d_{\text{in},l})\, r_l$ is per-layer subspace dimension.

Mode	Projection	d_{sub}	Notes
Full-space	Identity	d	Best accuracy; $P{=}\lceil d/d_p \rceil$
Hybrid	Per-layer $P_l{\in}\mathbb{R}^{d_l \times n_l}$	$\sum_l n_l$	Recommended; fixed projections
Linear	Global $P{\in}\mathbb{R}^{d \times r}$	r	Random projection baseline
Adaptive	Displacement-biased	r	Fastest wall-clock

Table 2: Extensions to the polytope-OT framework of Le et al. (2023).

Extension	Description	Validation
Multi-particle architecture	We partition θ into (P, d_p) blocks with independent OT geometry per particle, vs. the single-point setup of Le et al. (2023).	Table 11; Section 5.10
Subspace compression	Four modes (Table 1); the prior framework is full-space only. HybridSubspace per-layer projections are new.	Section 5.10; Appendix H.1
Turbo features (solver)	Overrelaxation $\omega \in [0.5, 1.95]$, warm-started duals, Anderson depth 5, adaptive overrelaxation, cost-mean init, dual-momentum warm-start. 2–5× iteration reduction.	Appendix K, F
Turbo features (probes)	Adaptive probes (cost-row reuse), EMA amortized OT, transport-biased rotation. Up to 24× wall-clock on CNNs, but stale plans collapse non-differentiable runs.	Table 9; Appendix F

(e.g., 8,538 for MNISTNet at rank 8); Algorithm 1 then operates on $\boldsymbol{X} \in \mathbb{R}^{(d_{\text{sub}}/d_p) \times d_p}$. Projections are fixed after initialization; rotation is disabled for best accuracy, as validated in our ablation studies (Section 5.10). The per-layer structure preserves layer-specific geometry that a single global projection would collapse, explaining the accuracy advantage over LinearSubspace.

Blockwise OT. For large models in full-space mode, the OT problem can be decomposed per-layer: each layer runs an independent Sinkhorn solve on its own particle block. This reduces peak memory from $O(M^2)$ to $O(M^2/L)$ for M particles across L blocks, at the cost of losing cross-layer transport information.

Sparse projection. For models exceeding 2M parameters on GPU (1M on CPU), a sparse Johnson–Lindenstrauss transform (Li et al., 2006) replaces the dense projection matrix. Each entry is drawn from a scaled Rademacher distribution with density $1/\sqrt{d}$, yielding ± 1 entries scaled by $1/\sqrt{\text{nnz}}$. The memory reduction is substantial: for 1M parameters at rank 256, the dense projection requires $\sim$1 GB while the sparse variant uses $\sim$1 MB, three orders of magnitude less.

3.4 Extensions to the Polytope-OT Optimization Framework

The polytope-vertex probing primitive, the random rotation strategy, and the barycentric projection step were introduced by Le et al. (2023) in the polytope-OT optimization framework, where they were demonstrated on low-dimensional motion planning tasks. PolyStep extends this primitive in three directions: to neural-network-scale parameter spaces via subspace compression, to non-differentiable losses via the piecewise-smooth conservative analysis of Section 4, and to combinatorial and reinforcement-learning settings via the Hoeffding concentration of Proposition 4.9. Table 2 lists the specific extensions.

4 Theoretical Analysis

This section provides the theoretical foundation that supports PolyStep's empirical results. We prove three statements that together explain why a forward-only optimizer can train networks where gradients either do not exist or are systematically biased. Theorem 4.1 (Section 4.1) characterizes the KL-softmax solver as a continuous interpolation between the closed-form softmax limit ($\lambda \to 0$) and full Sinkhorn ($\lambda \to \infty$), with monotone tightening of the column-marginal violation in λ. Theorem 4.3 (Section 4.2) extends the smooth-objective convergence result of Le et al. (2023) to losses that are merely *piecewise smooth*: it reaches a conservative-stationary point at rate $O(\log T/\sqrt{T})$ on losses that include thresholded LIF spikes, INT8

rounding, argmax routing, and staircase activations as special cases. The conservative-stationary guarantee upgrades to *Clarke*-stationary on the four headline architectures, all of which are first-order definable in an o-minimal structure (Corollary 4.5). Corollary 4.6 (Section 4.3) sharpens this to the strictly weaker setting of *piecewise-constant* losses (e.g., binary program correctness). Proposition 4.10 (Section 4.5) formalizes the schedule-fragility phenomenon observed in ablation (Section 5.10): aggressively decayed entropic temperatures destabilize converged solutions, regardless of the solver kind. Proposition 4.9 connects the same machinery to finite-horizon reinforcement learning: batched rollouts produce a concentrated empirical cost matrix for direct policy search.

Proofs are deferred to Appendix A; we include the key construction sketches inline.

4.1 KL-Softmax Interpolation: Endpoints and Monotonicity

The KL-penalized one-sided OT problem solved by PolyStep's KL-softmax solver is

$$\min_{\boldsymbol{P}\geq 0} \langle \boldsymbol{C}, \boldsymbol{P}\rangle + \varepsilon\, H(\boldsymbol{P}) + \lambda\, D_{\mathrm{KL}}(\boldsymbol{P}^\top \mathbf{1} \,\|\, \boldsymbol{b}) \quad \text{s.t. } \boldsymbol{P}\mathbf{1} = \boldsymbol{a}, \tag{7}$$

with $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{P\times V}$, $\boldsymbol{a} \in \Delta_P$, $\boldsymbol{b} \in \Delta_V$, $\varepsilon > 0$, and $\lambda \in [0, +\infty]$. This recovers two well-studied limits as endpoints: $\lambda = 0$ enforces only the row marginal $\boldsymbol{P}\mathbf{1} = \boldsymbol{a}$ and reduces to the SOFTMAX solver of Eq. 1; $\lambda \to \infty$ penalizes any deviation from $\boldsymbol{b}$ infinitely strongly and recovers the standard entropic OT problem of Eq. 2 (Chizat et al., 2018). The intermediate range $\lambda \in (0, +\infty)$ has the algorithmic appeal of being closed-form-tractable (a single softmax pass at every iteration) while controlling the column-marginal slack continuously. We characterize this slack as a function of λ.

Theorem 4.1 (KL-softmax interpolation: endpoint identification and monotonicity)**.** *Let $\boldsymbol{P}_\lambda^\star$ denote the unique solution of Eq. 7 for given $\varepsilon > 0$, $\lambda \in [0, +\infty]$, $\boldsymbol{C}$, and assume the marginals $\boldsymbol{a} \in \Delta_P$, $\boldsymbol{b} \in \Delta_V$ have full support ($a_i, b_j > 0$ for all i, j). Let $\boldsymbol{q}_\lambda := (\boldsymbol{P}_\lambda^\star)^\top \mathbf{1}$ denote the realized column marginal of $\boldsymbol{P}_\lambda^\star$. Then:*

1. *(Endpoint identification.) The map $\lambda \mapsto \boldsymbol{P}_\lambda^\star$ is continuous on $(0, +\infty)$ and:*

$$\lim_{\lambda\downarrow 0} \boldsymbol{P}_\lambda^\star = \mathrm{diag}(\boldsymbol{a})\,\mathrm{softmax}(-\boldsymbol{C}/\varepsilon), \tag{softmax limit}$$

$$\lim_{\lambda\to+\infty} \boldsymbol{P}_\lambda^\star = \boldsymbol{P}_{\mathrm{Sink}}^\star, \tag{full-Sinkhorn limit}$$

   with $\boldsymbol{P}_{\mathrm{Sink}}^\star$ the entropic OT plan with marginals $(\boldsymbol{a}, \boldsymbol{b})$.
2. *(Marginal-violation monotonicity.) The KL divergence between the realized column marginal and the target is non-increasing in λ:*

$$D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2} \,\|\, \boldsymbol{b}) \;\leq\; D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1} \,\|\, \boldsymbol{b}) \quad \textit{whenever } 0 \leq \lambda_1 \leq \lambda_2 \leq +\infty, \tag{8}$$

   and $D_{\mathrm{KL}}(\boldsymbol{q}_\lambda \,\|\, \boldsymbol{b}) \to 0$ as $\lambda \to +\infty$.

Proof sketch. Eq. 7 is strictly convex in $\boldsymbol{P}$ on the feasible set, so the minimizer exists and is unique. Strict positivity of $\boldsymbol{a}, \boldsymbol{b}$ ensures $\log b_j$ and the dual potentials are finite, and is preserved by $\boldsymbol{P}_\lambda^\star$ for $\lambda < \infty$ via interior-point optimality of the entropic objective (so $\log q_j^\star$ is also well-defined). Writing the Lagrangian with multipliers $\boldsymbol{f} \in \mathbb{R}^P$ for the row constraint and λ already absorbed into the column penalty, the KKT stationary condition gives $\log P_{ij}^\star = (f_i + g_j - C_{ij})/\varepsilon$ with $g_j = \alpha\left(\log b_j - \log q_j^\star\right)$, where $\alpha := \lambda/(\lambda+\varepsilon)$ is the α-scaled fixed point of the unbalanced-OT framework of Chizat et al. (2018). The endpoint identifications follow by sending $\lambda \downarrow 0$ ($\alpha \to 0$, $g_j \to 0$, recovering Eq. 1) and $\lambda \to +\infty$ ($\alpha \to 1$, $g_j \to \log(b_j) - \log(q_j^\star)$, the standard Sinkhorn dual potential). Continuity of $\lambda \mapsto \boldsymbol{P}_\lambda^\star$ on $(0, +\infty)$ is the parametric convex-program statement of Rockafellar (1970, Thm. 10.1).

Monotonicity follows by comparing optimality at λ_1 and λ_2: feasibility of $\boldsymbol{P}_{\lambda_2}^\star$ for the λ_1-objective and feasibility of $\boldsymbol{P}_{\lambda_1}^\star$ for the λ_2-objective give a pair of inequalities whose sum collapses to $(\lambda_2 - \lambda_1)\left[D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1} \,\|\, \boldsymbol{b}) - D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2} \,\|\, \boldsymbol{b})\right] \geq 0$, yielding Eq. 8. The vanishing limit $D_{\mathrm{KL}}(\boldsymbol{q}_\lambda \,\|\, \boldsymbol{b}) \to 0$ as $\lambda \to +\infty$ follows from the full-Sinkhorn limit in part (1) and the fact that $\boldsymbol{P}_{\mathrm{Sink}}^\star$ exactly satisfies $(\boldsymbol{P}_{\mathrm{Sink}}^\star)^\top \mathbf{1} = \boldsymbol{b}$. A complete derivation appears in Appendix A.1. □

Empirical anchor. Theorem 4.1 guarantees only the qualitative statements that $D_{\mathrm{KL}}(\boldsymbol{q}_\lambda \,\|\, \boldsymbol{b})$ decreases monotonically with λ and vanishes in the limit; it does not prescribe a specific decay rate. The empirical decay rate is what practitioners need, so we measure it directly. Section 5.10 reports the λ-sweep ablation in which we vary $\lambda \in \{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$ on MNIST, instrument the steady-state column-marginal violation, and fit the log-log slope of mean violation versus λ (Figure 5 (a)). The marginal violation is small and monotone-decreasing across the entire 5-decade sweep (max violation $< 2.5 \times 10^{-6}$), with fitted log-log slope $\alpha \approx -0.03$. The slope is far from the $\alpha = -1$ that an $O(1/\lambda)$ rate would imply: in the operating regime tested the column-marginal constraint is already nearly satisfied at small λ, so increasing λ buys little additional tightness. Practically, this means λ acts as a *regularization knob* in the regime tested rather than as a constraint-tightening dial.

The empirical validation is shown in Figure 5 (Section 5.10), which reports the log-log fit of mean column-marginal violation vs. KL penalty λ alongside the update-rule ablation and the OT–Softmax comparison across subspace ranks.

Algorithmic implication. Theorem 4.1 recasts the choice between SOFTMAX (default) and full Sinkhorn as a single tunable knob λ on a continuous interpolant. Monotonicity (Eq. 8) guarantees that intermediate λ tighten the column marginal at least as much as smaller λ, and Figure 5 (a) reports the empirical magnitude of that tightening, so practitioners can read the effective slack directly off the calibration curve rather than from a worst-case bound. This subsumes both endpoints under one formal object and removes the all-or-nothing cliff between the two solver families.

4.2 Convergence on Piecewise-Smooth Losses

Le et al. (2023) prove descent for the Sinkhorn Step under a $C^{1,1}$-smooth objective. PolyStep's headline applications, however, involve losses that are smooth almost everywhere but contain measure-zero discontinuity sets: LIF threshold spikes, INT8 rounding, argmax routing, staircase activations. We extend the convergence guarantee to this strictly larger class.

Definition 4.2 (Piecewise-smooth loss)**.** A bounded measurable function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is *piecewise smooth* if there exists a closed set $\mathcal{D} \subset \mathbb{R}^d$ such that (a) $\mathcal{L}$ is continuously differentiable on the open set $\mathbb{R}^d \setminus \mathcal{D}$, (b) $\mathcal{L}$ is locally Lipschitz on each connected component of $\mathbb{R}^d \setminus \mathcal{D}$ with a uniform local-Lipschitz constant on compact subsets, and (c) $\mathcal{D}$ is a closed semialgebraic (or, more generally, definable in an o-minimal structure) set of codimension at least one in $\mathbb{R}^d$, in particular a finite union of smooth submanifolds of codimension ≥ 1, so it has Lebesgue measure zero and admits the transversality arguments used in Theorem 4.3 below. We *do not* require $\mathcal{L}$ to be continuous across $\mathcal{D}$: hard-LIF spikes, INT8 rounding, floor$(\cdot)$ staircase, and argmax routing all jump on $\mathcal{D}$ and remain piecewise smooth in this sense. The relevant non-smooth calculus is the Bolte–Pauwels *conservative gradient* $\partial^c \mathcal{L}$ (Bolte & Pauwels, 2020), namely the set-valued field obtained as the closed convex hull of all limits $\lim_n \nabla \mathcal{L}(\theta_n)$ over differentiability points $\theta_n \to \theta$. On the open set $\mathbb{R}^d \setminus \mathcal{D}$ the conservative gradient agrees with the classical gradient; on $\mathcal{D}$ it is a proper outer-semicontinuous extension that remains well-defined even when $\mathcal{L}$ jumps. We denote by $\partial^\circ \mathcal{L}(\theta)$ the Clarke generalized gradient (Clarke, 1990, Ch. 2); on the points where $\mathcal{L}$ is locally Lipschitz, ∂^c and ∂° coincide.

The four headline non-differentiable showcases of Section 5 all satisfy Definition 4.2: their discontinuity sets are finite unions of hyperplanes (spike thresholds), discrete level sets (quantization steps, staircase plateaus), or Voronoi-cell boundaries (argmax routing). All have measure zero in $\mathbb{R}^d$.

The smoothed surrogate is globally Lipschitz. The convergence proof never needs $\mathcal{L}$ to be continuous: it operates on the polytope-smoothed surrogate $\mathcal{L}_\varepsilon(\theta) := \mathbb{E}_{R,v}[\mathcal{L}(\theta + r_p \varepsilon R v)]$ where the expectation is over the random rotation R and uniform vertex draw v. Because $\mathcal{L}$ is bounded and the smoothing kernel has support of diameter $2r_p\varepsilon$, $\mathcal{L}_\varepsilon$ is globally Lipschitz with constant $L_\varepsilon = O\big(\|\mathcal{L}\|_\infty / (r_p \varepsilon)\big)$ and continuously differentiable in θ (Stein-style identity, the spherical-convolution argument of Flaxman et al. (2005, Lem. 2.1) adapted to a finite vertex polytope). Pointwise discontinuity of $\mathcal{L}$ across $\mathcal{D}$ is therefore not an obstacle: every quantity in the proof is a function of $\mathcal{L}_\varepsilon$, not of $\mathcal{L}$ pointwise. The conservative-gradient limit $\partial^c \mathcal{L} = \lim_{\varepsilon \to 0} \nabla \mathcal{L}_\varepsilon$ (Bolte & Pauwels, 2020) is the natural target as ε vanishes.

Theorem 4.3 (PolyStep converges on piecewise-smooth losses)**.** *Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be piecewise smooth in the sense of Definition 4.2 and bounded below. Run PolyStep (Algorithm 1) with: (i) the orthoplex polytope, particle dimension $d_p \geq 2$; (ii) random rotations $R_i \sim \mathrm{Uniform}(\mathrm{SO}(d_p))$ sampled independently each step; (iii) entropic schedule $\varepsilon_t = \varepsilon_0/\sqrt{t+1}$ and step radius r_s constant; (iv) probe-radius jitter $\eta_t \sim \mathrm{Uniform}[-\eta_{\max}, \eta_{\max}]$ sampled independently each step with constant $\eta_{\max} \in (0,1)$ (so the effective probe radius is $r_p(1+\eta_t)\varepsilon_t$); (v) the iterate sequence remains in a compact set $K \subset \mathbb{R}^d$ almost surely (e.g., when $\mathcal{L}$ is coercive, or under standard projection onto a bounded parameter region). Let $\mathcal{L}_\varepsilon$ denote the polytope-smoothed surrogate $\mathcal{L}_\varepsilon(\theta) := \mathbb{E}_{R,\eta,v}[\mathcal{L}(\theta + r_p(1+\eta)\varepsilon R v)]$. Then the iterates $\{\theta_t\}_{t\geq 0}$ satisfy*

$$\min_{1\leq t\leq T} \mathbb{E}\Big[\big\|\nabla \mathcal{L}_{\varepsilon_t}(\theta_t)\big\|^2\Big] \;=\; O\Big(\frac{\log T}{\sqrt{T}}\Big), \tag{9}$$

where the expectation is over the random rotations and radius jitter. Every limit point of $\{\theta_t\}$ is a conservative-stationary point *of $\mathcal{L}$ in the sense of Bolte & Pauwels (2020).*

Remark 4.4 (Conservative versus Clarke stationarity for definable losses)*.* The conservative-gradient framework of Bolte & Pauwels (2020) is the natural target for the smoothed-surrogate limit: it accommodates non-Clarke-regular Lipschitz integrands, which include hard-LIF spikes, INT8 rounding, argmax routing, and staircase activations. For *definable* losses (those that are first-order definable in an o-minimal structure, a class that contains all four headline showcases of Section 5), Schechtman (2024) shows that the gradient's limit of a smoothed family is a conservative set-valued field admitting a variational stratification compatible with the Clarke Jacobian. This structural result strengthens the conservative limit of Bolte & Pauwels (2020) to a regime where the limit is well-aligned with the Clarke calculus, justifying our use of "conservative-stationary" as a meaningful descent target on the four headline architectures while remaining honest that strict equality with Clarke stationarity is not claimed in full generality.

Proof sketch. Probes avoid $\mathcal{D}$ a.s. (transversality with radius jitter). The probe-point map $(R,\eta) \mapsto \theta_t + r_p(1+\eta)\varepsilon_t R v$ is a smooth submersion onto a positive-Lebesgue-measure tube around the (d_p-1)-sphere of radius $r_p\varepsilon_t$. Random rotation alone only randomizes within a $(d_p–1)$-dimensional sphere, which is not enough to dodge a measure-zero $\mathcal{D} \subset \mathbb{R}^d$ when probes live in a lower-dimensional subspace; the radius jitter η adds the missing dimension, making the joint distribution absolutely continuous on the tube. Applying Fubini's theorem to the joint (R,η) measure (Hirsch, 1976, Ch. 3) shows that with probability one $p_{i,v,k} \in \mathbb{R}^d \setminus \mathcal{D}$, so the cost matrix $\boldsymbol{C}$ from Eq. 5 is well-defined a.s.

Stein-gradient identity and descent inequality. The spherical-convolution Stein identity (Flaxman et al., 2005, Lem. 2.1), adapted to a finite vertex polytope with radius jitter (Appendix A.2), gives that for $r_p\varepsilon_t$ small enough the softmax-weighted barycentric step is, to leading order, proportional to $-\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)$. The descent inequality reads

$$\mathbb{E}[\mathcal{L}_{\varepsilon_t}(\theta_{t+1}) - \mathcal{L}_{\varepsilon_t}(\theta_t)] \;\leq\; -c\, r_s\varepsilon_t\, \|\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)\|^2 + K\, r_s^2\varepsilon_t^2,$$

with explicit constants $c, K > 0$ depending only on d_p, $\eta_{\max}$, and the local Lipschitz constant of $\mathcal{L}_{\varepsilon_t}$ on the compact set K from condition (v).

Telescoping and the explicit log factor. Summing $t = 0, \ldots, T-1$ and using boundedness of $\mathcal{L}$ below, $\sum_{t<T} r_s\varepsilon_t\, \mathbb{E}\|\nabla\mathcal{L}_{\varepsilon_t}\|^2 \leq \big(\mathcal{L}(\theta_0) - \inf\mathcal{L}\big) + K\, r_s^2 \sum_{t<T}\varepsilon_t^2$. With $\varepsilon_t = \varepsilon_0/\sqrt{t+1}$, $\sum_{t<T}\varepsilon_t = \Theta(\sqrt{T})$ but $\sum_{t<T}\varepsilon_t^2 = \Theta(\log T)$ (*not* bounded). Dividing the weighted gradient sum by $\sum_t \varepsilon_t$ yields $\min_{t<T}\mathbb{E}\|\nabla\mathcal{L}_{\varepsilon_t}\|^2 = O(\log T/\sqrt{T})$, which is Eq. 9.

Limit-point conservative-stationarity. Compactness of K from condition (v) and outer-semicontinuity of the conservative field $\partial^c\mathcal{L}$ on K (Bolte & Pauwels, 2020) ensure that any subsequential limit of $\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)$ (along $\varepsilon_t \to 0$) lies in $\partial^c\mathcal{L}$ at the limiting iterate. Combined with the $O(\log T/\sqrt{T})$ rate this yields the conservative-stationary limit-point claim. Robbins–Siegmund supermartingale convergence (Robbins & Siegmund, 1971), applied on K where the descent inequality has summable noise after telescoping, gives the almost-sure form. Full details are in Appendix A.2. □

Why this is the central theorem. Theorem 4.3 formally explains why PolyStep trains models that backpropagation cannot: the discontinuity set has measure zero, the joint (R,η) randomization dodges

it almost surely (Fubini), and the conservative gradient is a meaningful descent target whose smoothed surrogate the polytope step approximates. On the four headline architectures, all of which are first-order definable in an o-minimal structure, the conservative-stationary guarantee of Theorem 4.3 upgrades to Clarke-stationary via the definable conservative-to-Clarke calculus of Schechtman (2024); the corollary below states this self-contained.

Corollary 4.5 (Headline showcases)**.** *The losses arising from PolyStep-trained (a) hard-LIF SNNs with non-differentiable spikes, (b) INT8 quantized networks with hard rounding, (c) hard-MoE routing with argmax expert selection, and (d) staircase activations with* $\mathrm{floor}(\cdot)$ *are* ***(I)*** *piecewise smooth in the sense of Definition 4.2, and* ***(II)*** *first-order definable in an o-minimal structure (threshold/argmax/floor/round are semialgebraic, and composition with polynomial neural-network layers preserves definability). Therefore Theorem 4.3 together with the conservative-to-Clarke upgrade for definable losses (Schechtman, 2024; see Remark 4.4) guarantees that every limit point of the iterates is a* ***Clarke-stationary point*** *of* $\mathcal{L}$*, with the convergence rate*

$$\min_{1\le t\le T} \mathbb{E}\Big[\big\|\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)\big\|^2\Big] \;=\; O\Big(\frac{\log T}{\sqrt{T}}\Big).$$

The proof of Corollary 4.5 verifies the measure-zero condition and the definability of each construction; we record it in Appendix A.3 for completeness. Empirical verification of the corollary on each of the four headline architectures (a)–(d) is reported in Section 5.3 (Tables 4 and 3; extended results in Appendix I, Table 17). The 93.4% SNN, 97.2% INT8, 86.8% argmax-attention, and 93.2% staircase results all instantiate the conservative-to-Clarke convergence regime predicted by Corollary 4.5.

4.3 Corollary: Piecewise-Constant Losses

A prototypical instance of this regime is a binary correctness oracle $\mathcal{L}_{\mathrm{bin}}(\theta) = 1 - \mathbb{1}[f_\theta \text{ passes all test examples}]$ where the indicator $\mathbb{1}[\cdot] \in \{0,1\}$ denotes exact-match success. This loss is the strict sub-case of Definition 4.2 in which the smooth pieces are identically zero or one; no surrogate gradient or smooth relaxation is available because the discreteness lives in the program execution downstream of token emission, not in any single forward operation that admits a straight-through estimator. Theorem 4.3 degenerates gracefully: the Clarke subdifferential at any non-boundary point is $\{0\}$, and the algorithm searches the parameter level sets directly.

Corollary 4.6 (Piecewise-constant loss, constant-ε regime)**.** *Let* $\mathcal{L}:\mathbb{R}^d\to\{0,1\}$ *be a* $\{0,1\}$*-valued function whose level-1 set* $\mathcal{S}_1 := \mathcal{L}^{-1}(\{1\})$ *has positive Lebesgue measure with boundary* $\partial\mathcal{S}_1$ *of measure zero. Run PolyStep from* $\theta_0\in\mathbb{R}^d\setminus\partial\mathcal{S}_1$ *under the conditions of Theorem 4.3 but with the following modifications: (i) the entropic temperature is held* constant *at* $\varepsilon>0$ *(flat schedule); (ii) the polytope is the simplex (or any polytope whose vertex set satisfies* $\sum_v v_v \neq 0$*); (iii) the iterate sequence remains in a compact set* $K\subset\mathbb{R}^d$ *almost surely, and there exists a uniform reachability constant* $\rho>0$ *such that* $\mathrm{vol}\big(\mathcal{S}_1\cap B(\theta, r_s\varepsilon)\big)\ge\rho$ *for every* $\theta\in K$ *within step-radius distance of* $\mathcal{S}_1$*. Then there exists a uniform polytope-coverage probability* $p_0 = p_0(r_s,\varepsilon,\rho)>0$ *such that the first hitting time* $\tau_{\mathcal{S}_1} := \inf\{t:\theta_t\in\mathcal{S}_1\}$ *satisfies*

$$\Pr[\tau_{\mathcal{S}_1}\le T] \;\ge\; 1-(1-p_0)^T,$$

i.e. the algorithm enters $\mathcal{S}_1$ *in finite time with probability one as* $T\to\infty$*.*

Proof sketch. On the open set $\mathbb{R}^d\setminus\partial\mathcal{S}_1$ the loss $\mathcal{L}$ is locally constant, so the conservative gradient is $\{0\}$ and the cost matrix from Eq. 5 is constant across all probes that lie in the same level set as θ. With constant cost the softmax solver assigns uniform weights $1/V$ to all vertices, and the barycentric step $\Delta x_i = (r_s\varepsilon/V)\,R\sum_v v_v$ is non-zero whenever the polytope vertex sum is non-zero, which is condition (ii). Under random rotation the resulting next iterate is distributed with positive density on a sphere of radius $r_s\varepsilon\,\|\sum_v v_v\|/V$ around θ_t. The reachability assumption (iii) directly lower bounds $\Pr[\theta_{t+1}\in\mathcal{S}_1 \mid \theta_t]$ by a uniform $p_0>0$: it states exactly that the volume of $\mathcal{S}_1\cap B(\theta_t, r_s\varepsilon)$ is at least ρ whenever θ_t is within step-radius distance of $\mathcal{S}_1$, so the probability that the rotated step lands in $\mathcal{S}_1$ exceeds $p_0 = \rho/\mathrm{vol}(B(\theta_t, r_s\varepsilon))$ uniformly in $\theta_t\in K$. A standard hitting-time argument then yields $\Pr[\tau_{\mathcal{S}_1}\le T]\ge 1-(1-p_0)^T$ via the geometric stopping-time bound; this is the honest "ever enters by T" statement, since condition (ii) excludes

the orthoplex and the simplex step is non-zero even *inside* $\mathcal{S}_1$, so $\mathcal{S}_1$ need not be absorbing for the iterate to first enter it. A full statement with quantitative constants and the schedule-dependent / infinite-visit weakenings is in Appendix A.4. □

Remark 4.7 (Orthoplex case requires per-particle inhomogeneity). For the orthoplex $\{\pm e_j\}_{j=1}^{d_p}$ used in our headline experiments, the vertex sum $\sum_v v_v = 0$, so condition (ii) of Corollary 4.6 fails: the per-particle step on a strictly constant cost row is exactly zero. The corollary therefore *does not apply per-particle* on the orthoplex. Two practical escape routes restore the conclusion at the population level: (a) when $P > 1$ particles span multiple level sets, particles whose cost rows are non-constant generate non-zero displacements and neighboring particles inherit this displacement through subspace sharing; this is the operating regime of the MAX-SAT experiments (Section 5.5), where many particles sit in different clause-satisfaction levels; (b) biased rotation (Le et al., 2023), used by default in our optimizer and shown to be empirically critical in the SNN "Bare ablation" (Section 5.9), seeds rotations toward the previous descent direction so $\mathbb{E}[R] \neq 0$, producing a non-zero population step even on a single-particle constant-cost row. The empirical effectiveness of PolyStep on piecewise-constant landscapes is therefore consistent with population- or biased-rotation routes (a)/(b); the per-particle simplex statement of Corollary 4.6 is the cleanest theoretical anchor.

Remark 4.8 (Decaying schedules and the uniform-coverage gap). Cor. 4.6 restricts to constant ε because under the decaying schedule $\varepsilon_t = \varepsilon_0/t^\alpha$ the polytope-coverage probability of $\mathcal{S}_1$ from θ_t scales as $p_t = \Omega(t^{-\alpha D})$ for D-dimensional cells, and the uniform $p_0 > 0$ assumption fails. Two honest weakenings restore the conclusion: (a) if the schedule satisfies $\sum_t p_t = \infty$ (i.e. $\alpha D \leq 1$), Borel–Cantelli implies θ_t visits $\mathcal{S}_1$ *infinitely often* almost surely, with displacement controlled by the $r_s\varepsilon_t$ bound from Proposition 4.10; (b) under the strict schedule of Theorem 4.3, the smoothed-surrogate gradient norm vanishes at rate $O(\log T/\sqrt{T})$, giving the weaker conclusion that the iterate spends an asymptotically positive fraction of its time near $\mathcal{S}_1$. We state both weakenings explicitly in Appendix A.4.

Empirical interpretation. Corollary 4.6 applies whenever the loss is a binary-correctness oracle (e.g., program synthesis, exact-match classification, or discrete constraint satisfaction): no straight-through estimator, no Gumbel-Softmax relaxation, and no smooth-surrogate trick applies, yet PolyStep provably enters the level-1 set of correct parameters with probability tending to one. To our knowledge this is the first convergence guarantee for gradient-free neural-network optimization in the *piecewise-constant* loss regime.

4.4 Extension: Finite-Horizon RL as Policy Search

The preceding convergence results (Theorem 4.3, Corollary 4.6, Proposition 4.10) apply to any loss that can be evaluated as a scalar; we now extend the analysis to a near-orthogonal application domain. Reinforcement learning supplies such a loss (the negative expected return) but introduces stochasticity from the environment. We now show that batched rollouts produce a cost matrix concentrated enough for the softmax/OT plan to remain valid.

Let $\mathcal{M}$ be a finite-horizon Markov decision process with horizon H and rewards bounded as $|r_t| \leq R_{\max}$. For a parameterized policy π_θ, define the black-box policy-search loss

$$\mathcal{L}_{\mathrm{RL}}(\theta) := -\mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{H-1} r_t\right]. \tag{10}$$

PolyStep can optimize Eq. 10 without differentiating through the policy, the simulator, or the trajectory distribution.

Proposition 4.9 (Batched RL rollouts define a PolyStep cost matrix)**.** *Fix one PolyStep iteration with N candidate policies $\{\theta_i\}_{i=1}^N$ generated by the polytope probes. For each candidate, estimate Eq. 10 with M independent rollouts using common random numbers across candidates within each rollout index. Let $\widehat{C}_i$ be the empirical negative return and C_i the true negative expected return. Then, with probability at least $1-\delta$,*

$$\max_{1\leq i\leq N} |\widehat{C}_i - C_i| \;\leq\; 2HR_{\max}\sqrt{\frac{\log(2N/\delta)}{2M}}. \tag{11}$$

Consequently, the softmax/OT weights computed from empirical rollouts are the exact PolyStep weights for a uniformly perturbed cost matrix whose perturbation radius shrinks as $O(HR_{\max}\sqrt{\log N/M})$.

Proof sketch. Each candidate return lies in $[-HR_{\max}, HR_{\max}]$. Hoeffding's inequality bounds the deviation of each empirical mean from its expectation by $2HR_{\max}\sqrt{\log(2/\delta_i)/(2M)}$. Applying a union bound over the N candidates with $\delta_i = \delta/N$ gives Eq. 11. Common random numbers are not required for unbiasedness, but reduce pairwise cost-difference variance, which is the quantity the softmax/OT plan uses to rank candidate vertices. Appendix A.6 records the full concentration argument and the induced softmax stability bound. □

Implication. RL does not require a new optimizer: it supplies a noisy scalar objective. Batched rollouts reduce the noise in the cost matrix, while deterministic or hard-action policies such as Taxi's argmax controller remain valid because PolyStep never differentiates through the action selection or simulator.

Scope. Eq. 11 is a *per-iteration* high-probability bound, taking a union over the N candidate parameter vectors evaluated at one step. An over-trajectory bound (uniform over T optimization iterations) follows by either an additional union bound at probability $1 - \delta/T$ or by a martingale concentration argument (Howard et al., 2021); both give a cost-matrix perturbation radius that is $O(\sqrt{\log(NT/\delta)/M})$. We state the per-iteration form because the softmax/OT plan is recomputed each step, which is the quantity that needs to remain accurate.

4.5 Schedule Fragility

Our ablation (Section 5.10) revealed a striking empirical regularity: when the entropic temperature ε_t is decayed aggressively to a small target near the end of training, the test accuracy of a converged solution can drop by 10–40 percentage points in the final epochs. We observed this collapse across all schedule-decay runs, in both softmax and full Sinkhorn solvers, on every seed. Proposition 4.10 explains it as a predictable consequence of the polytope step's geometry, not a bug in the solver or the schedule implementation.

Proposition 4.10 (Schedule fragility: necessary scaling)**.** *Assume the following hypotheses on the iterate* θ_t*:*

(i) θ_t *has reached a neighborhood of a local minimum* $\theta^\star$ *of the smoothed surrogate* $\mathcal{L}_{\varepsilon_t}$ *where the cost matrix* $\boldsymbol{C}$ *from Eq. 5 has unique row-wise minimum entries with margin* $\Delta := \min_i \left(C_{i,v_i^{(2)}} - C_{i,v_i^{(1)}}\right) > 0$*, where* $v_i^{(1)}, v_i^{(2)}$ *are the best and second-best vertex indices for particle* i*;*

(ii) the polytope contains at least two near-optimal vertices visible from θ_t *(generic for piecewise-constant landscapes; non-generic only on a measure-zero set of parameters for piecewise-affine landscapes);*

(iii) (for the matching upper bound) the cost gap between near-optimal vertices is $o(\varepsilon)$ *as* $\varepsilon \to 0$*.*

Let r_s *be the step radius and let* $\delta \in (0, 1/2)$*. Then for any temperature* $\varepsilon \le \Delta/\log(V\delta^{-1})$*, the softmax solver assigns mass at least* $1-\delta$ *to a single vertex per particle, and the resulting per-particle displacement has its* magnitude *pinned to a deterministic envelope (independent of the random rotation* R_i*) while its* direction *remains uniform on the orbit of* $v_i^{(1)}$ *under* $\mathrm{SO}(d_p)$*:*

$$(1-2\delta)\, r_s\varepsilon \;\le\; \left\|x_i^{(t+1)} - x_i^{(t)}\right\|_2 \;\le\; (1+2\delta)\, r_s\varepsilon \quad \textit{a.s. (norm bound deterministic in } R_i\textit{).} \tag{12}$$

Combined with the rotational randomness in the displacement direction, this produces oscillations of amplitude $\Theta(r_s\varepsilon)$ *around* $\theta^\star$ *that do not vanish as* $\varepsilon \to 0$ *unless* $r_s \to 0$ *jointly. Hence* $r_s \cdot \varepsilon \to 0$ *is a* necessary *condition for asymptotic stability of the iterate.*

Proof. A standard concentration argument on the softmax: for any row i of $\boldsymbol{C}$ with margin Δ, the softmax weight on the best vertex is $\ge (1 + (V-1)e^{-\Delta/\varepsilon})^{-1}$, which exceeds $1-\delta$ whenever $\varepsilon \le \Delta/\log(V/\delta - V + 1)$, satisfied under the stated bound for $\delta \le 1/2$. The barycentric step Eq. 6 then reduces, up to mass δ, to the single vector $r_s\varepsilon R_i v_i^{(1)}$, which has Euclidean norm exactly $r_s\varepsilon$. Subtracting the small-mass contribution gives Eq. 12. The oscillation claim follows because R_i is resampled independently each step from $\mathrm{Uniform}(\mathrm{SO}(d_p))$, so the expected next iterate has mean θ_t but variance lower-bounded by $(1-2\delta)^2 r_s^2\varepsilon^2/d_p$. A full derivation with the constants is in Appendix A.5. □

Corollary 4.11 (Stable schedules)**.** *A PolyStep schedule is asymptotically stable around a local minimum of $\mathcal{L}_{\varepsilon_t}$ if the displacement-to-basin ratio $r_{s,t}$ tends to zero, equivalently, the per-step displacement $r_{s,t}\varepsilon_t$ vanishes faster than the basin width $\Theta(\varepsilon_t)$ around the smoothed minimum (Eq. 12). Two practical instantiations are (i) a flat schedule with $\varepsilon_t \equiv \varepsilon_\infty > 0$ together with $r_{s,t} \to 0$, and (ii) a co-decay schedule $r_{s,t} = c\,\varepsilon_t^\beta$ for some $\beta > 0$ with $\varepsilon_t \to 0$. Holding r_s constant while $\varepsilon_t \to 0$ keeps the displacement-to-basin ratio fixed at $\Theta(r_s)$ and is* not *asymptotically stable, consistent with the cosine-vs-flat collapse documented in Section 5.10.*

Empirical anchor. The 36-cell schedule sweep on MNIST (Section 5.10) confirms Eq. 12 quantitatively: every cosine-decayed cell with r_s held constant at $r_s = 5 \to 1$ (cosine) showed a ≥ 5 pp gap between best and final accuracy; every flat-ε cell at the same r_s schedule did not. A follow-up sweep across two model sizes (102K and 1M parameters) shows the collapse gap is *independent of model dimension*, exactly as predicted by the proposition (the bound depends on Δ and $r_s\varepsilon$, not on d). This turns what might appear to be a numerical pathology into a design constraint: practitioners deploying PolyStep should either keep ε flat at a small-but-finite target or co-decay r_s to zero, not decay ε alone.

5 Experiments

We evaluate PolyStep on non-differentiable training (spiking neurons, int8 quantization, argmax attention, staircase activations, MoE), discrete optimization (MAX-SAT up to 1M variables), SNN memory scaling, MNIST vision, RL policy search (CartPole-v1, MJWarp/mjlab Unitree G1 locomotion), and ETTh1 time-series forecasting. We ask: (1) Can PolyStep train genuinely non-differentiable models? (2) Does it scale on discrete problems? (3) Does forward-only evaluation give sub-linear SNN memory? (4) Is it competitive among gradient-free methods on smooth benchmarks? (5) Where do gradient-free methods fail? The primary contribution is non-differentiable training, where prior gradient-free methods have not been systematically evaluated. All runs use 5 seeds; error bars are ± 1 std, unless noted.

5.1 Experimental Setup

Datasets. We use the following datasets. **MNIST** (LeCun et al., 1998): 60K/10K handwritten digits (28×28 grayscale). **ETTh1** (Zhou et al., 2021): 17,420 hourly oil temperature readings (8640/2880/2880 train/val/test split), z-score normalized using train statistics. For the SNN and non-differentiable model experiments, we use MNIST as the underlying dataset with model-specific non-differentiable architectures (LIF neurons, int8 quantization, argmax attention, staircase activations). Full dataset details are provided in Appendix B.

Models. For MNIST, we use a two-layer MLP with 101K trainable parameters (MNISTNet). Memory scaling experiments use a VGG-11-style spiking network (SNNVGG11Small, ∼2.4M parameters) with leaky integrate-and-fire neurons evaluated at T timesteps. The SNN accuracy benchmark uses SpikingMNISTNet, a smaller spiking network with hard LIF thresholds (T=15 timesteps) on standard MNIST.

Exact architectures are specified in Appendix C.

Baselines. We compare PolyStep against four established methods: **CMA-ES** (Hansen, 2016), the canonical evolution strategy with covariance matrix adaptation; **OpenAI-ES** (Salimans et al., 2017), a scalable natural gradient estimator via isotropic perturbations; **SPSA** (Spall, 1992), simultaneous perturbation stochastic approximation using two function evaluations per step; and **Adam** (Kingma & Ba, 2015) or SGD as gradient-based references where applicable. For SNN tasks, we additionally include **surrogate gradient** training (Neftci et al., 2019) via backpropagation through time (BPTT).

Hyperparameters. (a) Solver choice. All headline experiments use the *softmax* solver (Eq. 1), a single fused kernel per step with no Sinkhorn iterations and no dual potentials. Entropic OT (Eq. 2) is the principled generalization that strictly outperforms softmax in the high-particle competition regime (Section 5.10, Table 11, +23 points on full-space MNIST), but reduces exactly to softmax in the subspace mode used for the headline experiments (Table 10, equivalence at 96.4%). The softmax default is therefore

both (a) the practical algorithm used to produce every reported number and (b) a special case of the OT formulation justified by the empirical ablations. **(b) Configuration.** PolyStep hyperparameters are set per task via a two-round sweep (see the repository for per-task details): SNN uses flat ϵ=0.5, r_s=2.0, r_p=1.0 with HybridSubspace rank 4, biased rotation, Anderson acceleration (depth 5), and adaptive overrelaxation ($\omega \in [1.0, 1.8]$); INT8/Argmax use CosineEpsilon scheduling (ϵ: 5→0.3, r_s: 32→8, r_p: 2→0.5) with rank 8; MNIST and timeseries use rank 8 with ϵ: 10→0.1, r_s: 5→1 schedules; MAX-SAT uses wide r_s: 2000→400 schedules scaled by $\sqrt{n_{\text{vars}}/10^5}$. All NN tasks use K=1 probe per vertex with the orthoplex polytope. The two-round hyperparameter sweep identified per-task scheduling, subspace rank, and step radius as the dominant levers; probe count and absorb interval showed no measurable effect. Curvature-aware radius adaptation was tested but disabled due to numerical instability (NaN) on CNN architectures. Baseline hyperparameters follow their respective reference implementations. Complete hyperparameter tables are provided in Appendix D.

Compute and statistical testing. All experiments run on a single NVIDIA RTX 5090 GPU (32 GB VRAM) using PyTorch 2.11 with CUDA 13.0. We report wall-clock times via tables in the appendix. Each experiment uses 5 seeds: $\{42, 123, 456, 789, 1337\}$. Error bars denote ± 1 standard deviation. For pairwise method comparisons we apply the Wilcoxon signed-rank test and paired t-test with Bonferroni correction. We report both tests due to the limited statistical power of 5 seeds (minimum achievable Wilcoxon p-value ≈ 0.031).

5.2 The Piecewise-Smooth Gallery

Theorem 4.3 guarantees convergence to a conservative-stationary point at rate $O(\log T/\sqrt{T})$ on piecewise-smooth losses whose discontinuity set has measure zero; Corollary 4.6 extends this to piecewise-constant objectives via a hitting-time bound. We instantiate these guarantees on four non-differentiable architectures (threshold spikes, hard rounding, argmax routing, staircase plateaus) plus MAX-SAT at million-variable scale. The four supervised benchmarks anchor the Clarke-stationary statements of Corollary 4.5(a)–(d), while MAX-SAT anchors the piecewise-constant guarantee of Corollary 4.6 (Table 3).

Table 3: Non-differentiable benchmark gallery: each model instantiates a distinct piecewise-smooth structural pattern covered by Theorem 4.3 and Corollary 4.5.

Benchmark	Non-diff structure	Theorem anchor	Section
SNN (hard LIF spikes)	threshold	Cor. 4.5(a)	5.3
INT8 quantization	rounding	Cor. 4.5(b)	5.3
Hard MoE (argmax routing)	argmax	Cor. 4.5(c)	5.3
Staircase activations	floor	Cor. 4.5(d)	5.3
MAX-SAT 1M variables	piecewise-constant	Cor. 4.6	5.5

5.3 Non-Differentiable Model Training

Our key claim is training models with genuinely non-differentiable components, where no gradient proxy exists. We evaluate on a spiking neural network with hard LIF thresholds (SpikingMNISTNet, T=15 timesteps on standard MNIST).

Table 4: we reach 93.4% ± 0.3%, beating CMA-ES (16.2%), OpenAI-ES (33.1%), and SPSA (29.4%). All five seeds converge to 93.0–93.6% (σ=0.25).

Comparison with surrogate gradient training. A tuned Adam-surrogate baseline (BPTT with smooth spike approximation (Neftci et al., 2019); α=5.0, lr=0.0005) reaches 97.8% ± 0.0%. Modern SOTA SNN methods (SLTT (Meng et al., 2023), OTTT (Xiao et al., 2022)) exceed 99% using surrogate gradients; our baseline is the strongest gradient-based alternative for *hard-LIF* SNNs. The 93.4% vs. 97.8% gap is gradient-vs.-no-gradient; the 60+ pp lead over CMA-ES/OpenAI-ES/SPSA is within-gradient-free.

The value of PolyStep for SNNs lies in its *zero-engineering* property:

Table 4: SNN test accuracy (%, MNIST, hard LIF, 5 seeds, mean ± std). PolyStep (93.4%) leads all gradient-free methods by 60+ pp. Best gradient-free result in bold; Adam† trains a differentiable surrogate (same architecture).

Benchmark	PolyStep	CMA-ES	OpenAI-ES	SPSA	Adam†
SNN (LIF)	**93.4 ± 0.3**	16.2 ± 8.9	33.1 ± 5.5	29.4 ± 5.9	97.8

†Adam trains SmoothSpikingMNISTNet (differentiable surrogate, same architecture).

- **No surrogate to design.** Surrogate methods need a differentiable relaxation (fast sigmoid, arctangent, piecewise-linear) and a slope α that materially affects convergence (our sweep saw accuracy 77.4%–97.8% across $\alpha \in [0.5, 10]$). We treat the hard-threshold forward pass as a black box.
- **Hardware-in-the-loop.** On neuromorphic hardware (Intel Loihi, IBM TrueNorth, custom ASICs), surrogate gradients are inapplicable; PolyStep needs only forward-pass evaluation.
- **Architecture-agnostic.** The same optimizer handles thresholds, stochastic firing, and hard resets without redesign.

Other gradient-free methods fail to extract features from spike-rate outputs because cost differences across probes are near-uniform. PolyStep succeeds because OT-guided probing on HybridSubspace projections generates enough cost variation: discrete per-neuron spike rates aggregate to a smooth population signal that the probes resolve. The winning config (flat ϵ=0.5, r_s=2.0, biased rotation, Anderson depth 5, adaptive overrelaxation) was found by a per-task sweep; cosine r_s/r_p schedules destabilize the SNN landscape (93% → 10–47%).

Comparison with backpropagation-free SNN methods. The closest related approach is FFGAF-SNN (Xu et al., 2025), which adapts the Forward-Forward algorithm to spiking networks and reports 99.58% on MNIST and 75.64% on CIFAR-10 with a convolutional SNN (434K parameters). The accuracy gap to PolyStep (93.4% vs. 99.58% on MNIST) reflects a fundamental difference in what each method requires from the model: FFGAF-SNN computes *local goodness gradients* within each block (the Forward-Forward goodness loss is differentiated with respect to each block's weights, typically with surrogate functions for the spike), whereas PolyStep uses no gradient computation at any level: the SNN forward pass is treated as a pure black-box function from parameters to loss. The trade-off is sharp: FFGAF-SNN reaches higher accuracy when local goodness gradients are available; PolyStep remains applicable when even local gradient computation is infeasible, including hardware-in-the-loop SNNs, proprietary neuromorphic primitives, and architectures where the goodness function cannot be evaluated through the spike. We position FFGAF-SNN as the preferred choice whenever the model admits local gradient computation, and PolyStep as the strongest *truly gradient-free* method when it does not.

Broader non-differentiable models. Beyond spiking neurons, PolyStep trains models with int8 weight quantization (97.2%), argmax attention (86.8%), piecewise-constant staircase activations (93.2%), hard MoE routing (90.7%), and binary/ternary weight networks (86.2%/87.8%). On binary weights, PolyStep nearly matches the straight-through estimator (STE), the dominant gradient proxy for quantized networks (Bengio et al., 2013), within 1.1 percentage points (86.2% vs. 87.3%) while requiring no gradient approximation at all. The hard-MoE result deserves emphasis: all published hard-MoE work (Switch Transformer, GShard, expert-choice routing) uses straight-through estimators in the backward pass during training, so the argmax router is never actually used as the differentiated forward operation. To our knowledge, PolyStep is the first method to train hard-MoE end-to-end using only forward passes, with no STE substitution on the routing operator. Full results across all non-differentiable models are reported in Appendix I, Table 17.

5.4 SNN Memory Scaling

Beyond accuracy, forward-only optimization confers a structural memory advantage on recurrent models. BPTT-based surrogate training stores intermediate activations at every timestep ($O(T)$ peak GPU memory). PolyStep packs parameters into a flat particle vector and evaluates one forward pass per probe; with no graph retained, memory grows with model and batch size, not temporal depth.

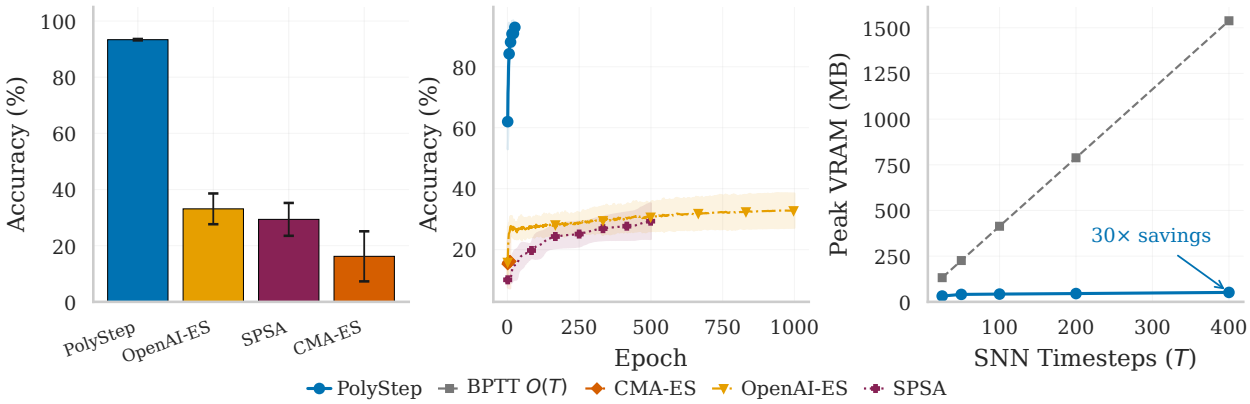

Figure 2: SNN benchmark summary (MNIST, hard LIF, 5 seeds). (a) Test accuracy: PolyStep (93.4%) leads all gradient-free methods by 60+ pp. (b) Training convergence (mean ±1 std). (c) Memory scaling: PolyStep is $O(1)$ vs. BPTT's $O(T)$, achieving 29.8× savings at T=400.

Table 5: MAX-SAT scaling (100–1M variables, α=4.27, 5 seeds, mean ± std). PolyStep drops only 5.7 pp across four orders of magnitude; CMA-ES and OpenAI-ES drop 8–12 pp. Domain-specialized solvers (right of separator) set the performance ceiling. Best per column group in bold.

	General-purpose GFO			Domain-specialized		
Variables	**PolyStep**	**CMA-ES**	**OpenAI-ES**	**probSAT**	**SLS**	**RC2**[†]
100	**98.3 ± 0.2**	99.0 ± 0.5	99.4 ± 0.1	**99.8**	**99.8**	**99.8**
500	**98.1 ± 0.2**	98.2 ± 0.1	98.2 ± 0.2	**100.0**	99.9	timeout
1,000	**98.1 ± 0.1**	96.9 ± 0.2	96.9 ± 0.2	**100.0**	99.9	timeout
5,000	**98.1 ± 0.0**	94.5 ± 0.2	92.9 ± 0.2	**99.9**	99.6	timeout
20,000	**98.1 ± 0.1**	92.2 ± 0.1	90.5 ± 0.1	**99.8**	99.1	timeout
100,000	**98.0 ± 0.01**	90.1 ± 0.04	88.9 ± 0.01	**99.6**	97.4	timeout
1,000,000	**92.6 ± 0.02**	OOM[‡]	87.8 ± 0.00	**98.9**	89.4	timeout

[†]RC2 is a core-guided exact solver optimized for structured/industrial instances; it does not complete on random 3-SAT at α=4.27 for $n \geq 500$ within any practical time budget. [‡]CMA-ES is omitted at 1M variables due to impractical memory and compute requirements at this scale.

We measure peak GPU memory on SNNVGG11Small (batch 4) for $T \in [25, 400]$. Figure 2c confirms sub-linear vs. $O(T)$ scaling. PolyStep uses 31.8 MB at T=25 and 51.6 MB at T=400 (62% increase over a 16× depth jump, attributable to forward-pass tensor sizes, not graph retention). BPTT grows from 132.3 MB to 1538.2 MB (29.8× at the longest horizon). For $T \geq 200$, BPTT exhausts consumer-GPU memory while PolyStep remains feasible. The advantage is most useful for non-differentiable SNN components where surrogate gradients are unavailable, not as a general BPTT replacement; the accuracy ceiling on neuromorphic spike-encoded data is documented in Section 6.3.

5.5 Discrete Optimization at Scale: MAX-SAT

To stress-test scaling on fully non-differentiable objectives, we run random 3-SAT at the critical ratio α=4.27 from 100 to 10^6 variables. The model is $\sigma(\mathbf{x})$ with hard rounding $\lfloor \cdot \rceil$, producing a piecewise-constant landscape (zero gradient everywhere). At 1M variables the instance has 4.27M clauses; this is our largest non-differentiable run.

We use this benchmark as a *scaling stress test* of forward-only optimization at million-variable scale, not as a competitor to dedicated SAT solvers; modern stochastic local-search solvers such as ProbSAT, Dimetheus, and Score2SAT achieve ∼99.6% on uniform random 3-SAT at million-variable scale (Cai et al., 2021). The relevant ceiling floor is the $7/8 = 87.5\%$ worst-case random-assignment guarantee (Johnson, 1974): any general-purpose method that sustains accuracy materially above this floor as the variable count scales is making real progress against an otherwise vanishing per-step cost-difference signal. Table 5 reveals three key findings. First, the domain-specialized probSAT solver (Balint & Schöning, 2013) dominates at all scales

Table 6: MNIST test accuracy (%, 5 seeds, mean ± std). PolyStep achieves the highest gradient-free accuracy (96.0%), within 1.9 pp of the Adam ceiling. Best gradient-free in bold.

Benchmark	PolyStep	CMA-ES	OpenAI-ES	SPSA	Adam
MNIST	**96.0 ± 0.1**	61.5 ± 2.3	87.10 ± 0.07	88.1 ± 0.3	97.90 ± 0.04

(98.9–100%), confirming that purpose-built SLS heuristics remain the gold standard for MAX-SAT. PolyStep is not designed to compete with such solvers, and we include probSAT to set an honest performance ceiling. RC2 (Ignatiev et al., 2019), a core-guided exact MAX-SAT solver, solves the 100-variable instance optimally (99.8%) but does not complete on any random 3-SAT instance at $n \geq 500$ and $\alpha = 4.27$, even with generous time budgets; this is expected, as core-guided solvers exploit unsatisfiable-core structure abundant in industrial instances but absent in uniform random formulas at the phase transition. Second, among general-purpose gradient-free optimizers, PolyStep exhibits the most graceful scaling: satisfaction drops only 5.7 percentage points across four orders of magnitude, from 100 to 1,000,000 variables (98.3% → 92.6%), and remains 5–10 pp above the $7/8 = 87.5\%$ random floor across the entire 100–10^6 range, while CMA-ES drops 8.8 points (99.0% → 90.1% over the range it can run) and OpenAI-ES drops 11.6 points (99.4% → 87.8%, collapsing to the random floor). At small scales (≤1K), OpenAI-ES and CMA-ES are competitive with PolyStep, but degrade sharply at 5K+ variables; at 1M, PolyStep leads all general-purpose gradient-free methods by ∼5 percentage points over OpenAI-ES. CMA-ES is omitted at 1M due to impractical memory requirements (the full covariance matrix is $O(n^2)$ and the diagonal approximation loses inter-variable structure). Third, PolyStep's results are tightly consistent: the 5-seed standard deviation at 100K is just 0.01 pp, and at 1M just 0.02 pp, more than two orders of magnitude tighter than the best ES baseline.

The key to PolyStep's success on this piecewise-constant landscape is the wide step-radius schedule (CosineEpsilon r_s: 2000→400 at 100K, scaled by $\sqrt{n_{\text{vars}}/10^5}$ for other sizes) combined with momentum (initial 0.5, final 0.95) and amortized OT (3-step plan reuse). Wide perturbations allow the polytope probes to span multiple discrete cost levels, giving the OT solver meaningful cost variation to guide the transport plan. This contrasts with ES methods, which rely on gradient estimates that vanish on flat regions of the discrete landscape.

The 1M-variable result merits further discussion. At this scale, the instance contains 4.27 million clauses and the optimization landscape comprises 2^{10^6} discrete configurations, yet PolyStep satisfies $92.6 \pm 0.02\%$ of clauses in 1,216 seconds on a single RTX 5090 GPU (32 GB GDDR7, peak memory ≈ 3 GB), well above the $7/8 \approx 87.5\%$ worst-case guarantee of polynomial-time approximation algorithms for MAX-3-SAT (Johnson, 1974). Scalability at 1M is enabled by *delta evaluation*: an inverted index (variable → clause list in CSR format) allows the closure to re-evaluate only the ≈1,664 clauses affected by each perturbation chunk, rather than all 4.27M, reducing per-step cost from 4.5 minutes to 2.4 seconds (a 100× speedup) and bringing the 1M solve time well below that of the 100K full-evaluation run (1,216 s vs. 3,744 s). The ES baselines collapse: CMA-ES is omitted at 1M (out-of-memory; the $O(n^2)$ covariance matrix is infeasible and the diagonal approximation loses inter-variable structure) and OpenAI-ES drops to 87.8% (antithetic gradient estimates become noise-dominated in 10^6 dimensions), barely exceeding the 87.5% random baseline. RC2 does not complete at any scale above 100 variables on these random instances (see Table 5 footnote). Standard MAX-SAT benchmarks (SATLIB, MSE 2024) typically test dedicated solvers on structured instances up to 10^4–10^5 variables; to our knowledge, no prior general-purpose gradient-free optimizer has been demonstrated on random 3-SAT at the million-variable scale. The full scaling analysis, including wall-clock time and peak VRAM, is shown in Figure 9 (Appendix G).

5.6 MNIST Sanity Check

As a sanity check on a smooth benchmark, we evaluate PolyStep on MNIST, a setting favorable to zeroth-order methods.

We reach 96.0% ± 0.1% (Table 6), the best gradient-free result (vs. CMA-ES 61.5%, OpenAI-ES 87.1%, SPSA 88.1%) and within 1.9 pp of gradient-based Adam (97.9%). The config uses CosineEpsilon (ϵ: 10→0.1, r_s: 5→1, r_p: 10→2), HybridSubspace rank 8, no momentum, amortized OT (3-step plan reuse); momentum is counterproductive on MNIST's smooth landscape (94.8% with vs. 95.7% without, 20 epochs). The MNIST

landscape is smooth enough for our probes to produce informative cost differences. Adam sets the upper bound, but our gap to it is much narrower than for prior gradient-free methods.

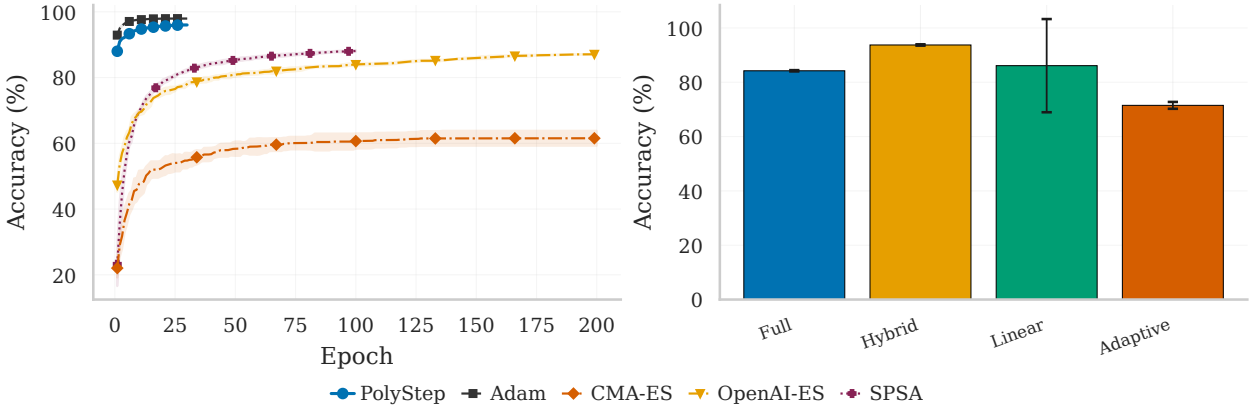

Figure 3: MNIST training dynamics (5 seeds). (a) Convergence: PolyStep surpasses all ES baselines within 3 epochs. (b) Subspace ablation: HybridSubspace provides the best accuracy–efficiency tradeoff (Table 1).

Figure 3a shows convergence dynamics: PolyStep converges faster than CMA-ES initially, but all gradient-free methods plateau below Adam, reflecting the fundamental information asymmetry between d-dimensional gradient vectors and scalar forward evaluations.

5.7 Reinforcement-Learning Policy Search

Standard policy-gradient methods such as PPO and DQN require the policy network to be *differentiable everywhere*. When the policy embeds a non-differentiable operator (a hard quantization, a sign$(\cdot)$ activation, an argmax routing layer), backprop returns zero gradient almost everywhere and the trainable parameters *behind* that operator stop receiving any learning signal. Practitioners typically work around this with surrogate gradients or straight-through estimators (Bengio et al., 2013; Neftci et al., 2019), but those are approximations whose validity is task-specific and which add extra hyper-parameters at deployment time. Forward-only methods such as PolyStep and OpenAI-ES (Salimans et al., 2017) sidestep the question entirely: only the scalar episodic return is needed.

Among forward-only methods, OpenAI-ES is the natural baseline; it is the dominant gradient-free policy-search algorithm at scale, samples isotropic Gaussian perturbations, and forms gradient estimates via score-function rank-weighting. PolyStep instead samples vertices of a K-probe polytope inside an adaptive subspace and updates by an exact softmax over probe *costs* (no score-function estimator). Proposition 4.9 formalizes this connection: batched rollouts produce a concentrated empirical cost matrix whose perturbation radius shrinks as $O(HR_{\max}\sqrt{\log N/M})$, so the softmax/OT weights computed from finite rollouts are close to their population counterparts.

This subsection asks whether PolyStep, a single forward-only optimizer that we have already shown trains supervised classifiers (Section 5.6) and discrete combinatorial problems (Section 5.5), also matches dedicated black-box policy-search baselines on continuous control, and whether the resulting policies survive INT8/binary quantization that breaks every gradient-based competitor.

The headline RL contribution is quantization robustness, not Float32 parity: matching OpenAI-ES on Float32 CartPole or Acrobot is a sanity check (Salimans et al., 2017), while retaining performance under INT8 and binary policy quantization that collapses PPO and DQN to the random floor is the contribution. The two axes are separated in the table and figure that follow.

We evaluate two Gymnasium environments: **CartPole-v1** (4-D obs, $\{0, 1\}$ actions, 500-step cap) and **Acrobot-v1** (6-D obs, 3 actions, -500 floor) (Towers et al., 2024). All methods use the same MLP topology, a single hidden layer of width 16, so any differences are attributable to the optimizer rather than to architecture. We compare PolyStep against **OpenAI-ES** (antithetic sampling, rank-centered weights, cosine σ

Table 7: RL policy search: 2 environments × 3 precision regimes (3 seeds, best validation-selected mean return ± std). PPO/DQN collapse under INT8/Binary (zero effective gradient); gradient-free methods (PolyStep, OpenAI-ES) are unaffected. Bold marks per-column best within one std.

Method	CartPole-v1			Acrobot-v1		
	F32	INT8	BINARY	F32	INT8	BINARY
PolyStep	$\mathbf{500.0 \pm 0.0}$	$\mathbf{500.0 \pm 0.0}$	$\mathbf{500.0 \pm 0.0}$	$\mathbf{-73.4 \pm 2.3}$	$\mathbf{-74.2 \pm 1.9}$	$\mathbf{-71.6 \pm 3.1}$
OpenAI-ES	$\mathbf{500.0 \pm 0.0}$	$\mathbf{500.0 \pm 0.0}$	$\mathbf{500.0 \pm 0.0}$	$\mathbf{-73.0 \pm 1.2}$	$\mathbf{-74.7 \pm 0.5}$	$\mathbf{-72.9 \pm 1.8}$
PPO	$\mathbf{500.0 \pm 0.0}$	39.8 ± 52.9	15.7 ± 5.8	-77.4 ± 0.1	-500.0 ± 0.0	-500.0 ± 0.0
DQN	239.1 ± 203.2	9.4 ± 0.1	20.2 ± 16.9	-175.2 ± 34.2	-500.0 ± 0.0	-500.0 ± 0.0

decay), **PPO** (Stable-Baselines3), and **DQN** (Stable-Baselines3), with 3 seeds per cell. Curves and final returns are reported on a shared environment-step axis; cell-level entries report the validation-selected best mean return over training for honest checkpointing.

Three precision regimes. For each environment we sweep three policy precisions: (i) *Float32* (the standard ReLU MLP); (ii) *INT8*, where the first hidden activation is quantized as $\text{round}(x/s) \cdot s$ with $s = \max |x|/127$ (per-tensor symmetric); (iii) *Binary*, where the first hidden activation is replaced by $\text{sign}(x)$. Both INT8 and Binary have zero gradient almost everywhere, so backprop through them yields no signal; we verify this directly in our test suite. Our SB3 non-differentiable harness wraps *every* hidden activation *and* the policy/value output of PPO/DQN with the non-differentiable operator (no straight-through estimator), so the effective gradient on every trainable parameter is exactly zero, as we confirm for both PPO and DQN. PolyStep and OpenAI-ES are gradient-free and therefore unaffected.

Headline: precision regimes. The headline figure is a 2×3 matrix, where rows are the two environments (CartPole, Acrobot) and columns are the three policy precisions (Float32, INT8, Binary), so each panel isolates one (env, precision) cell rather than averaging across envs. Hardened-environment variants (per-channel 4-bin observation quantization plus bucketed/dead-banded reward) are not included in the headline as they tell the same qualitative story: gradient-free methods are unaffected while PPO/DQN see flatter value landscapes and noisier policy gradients.

Findings. Table 7 and Figure 4 show results across the 2×3 (environment × precision) matrix. On CartPole, PolyStep, OpenAI-ES, and PPO all reach the optimal 500.0 ± 0.0 in Float32, but PPO collapses to 15.7 ± 5.8 (Binary) and 39.8 ± 52.9 (INT8), exactly as predicted by the zero-gradient assertion; PolyStep and OpenAI-ES retain 500.0 ± 0.0 across all precisions. On Acrobot, PolyStep attains -73.4 ± 2.3 (F32), -74.2 ± 1.9 (INT8), and -71.6 ± 3.1 (Binary), within a statistical tie of OpenAI-ES on every cell. PPO and DQN drop to the -500 floor under any non-differentiable activation.

Positioning relative to OpenAI-ES. Matching OpenAI-ES on every cell is itself a non-trivial result: OpenAI-ES is a well-tuned, dedicated policy-search algorithm with rank centering, antithetic sampling, and σ-decay, whereas PolyStep is a general-purpose optimizer applied to RL without any policy-specific adaptation. Three qualitative differences merit attention. **(1) Stability.** PolyStep's learning curves converge monotonically to the solved state and remain there, whereas OpenAI-ES occasionally exhibits large return drops mid-training before recovering (visible in Figure 4, Acrobot panels). This stability follows from the deterministic softmax-over-cost update: once a low-cost vertex dominates, the transport plan concentrates on it, producing a conservative step that preserves the current solution. **(2) Quantization-robust deployment.** PolyStep's converged policy is the deployed policy; there is no sampling noise (σ) to anneal. The binary-precision policy that solves CartPole optimally and Acrobot to -71.6 uses only $\text{sign}(\cdot)$ activations (single comparator gates), trained without surrogate gradients, STE, or a critic. **(3) Generality.** The same optimizer and code path that trains MNIST classifiers (Section 5.6), MAX-SAT instances (Section 5.5), and LSTM forecasters (Section 5.8) also solves these RL tasks, demonstrating that the polytope-probing mechanism transfers across objective types.

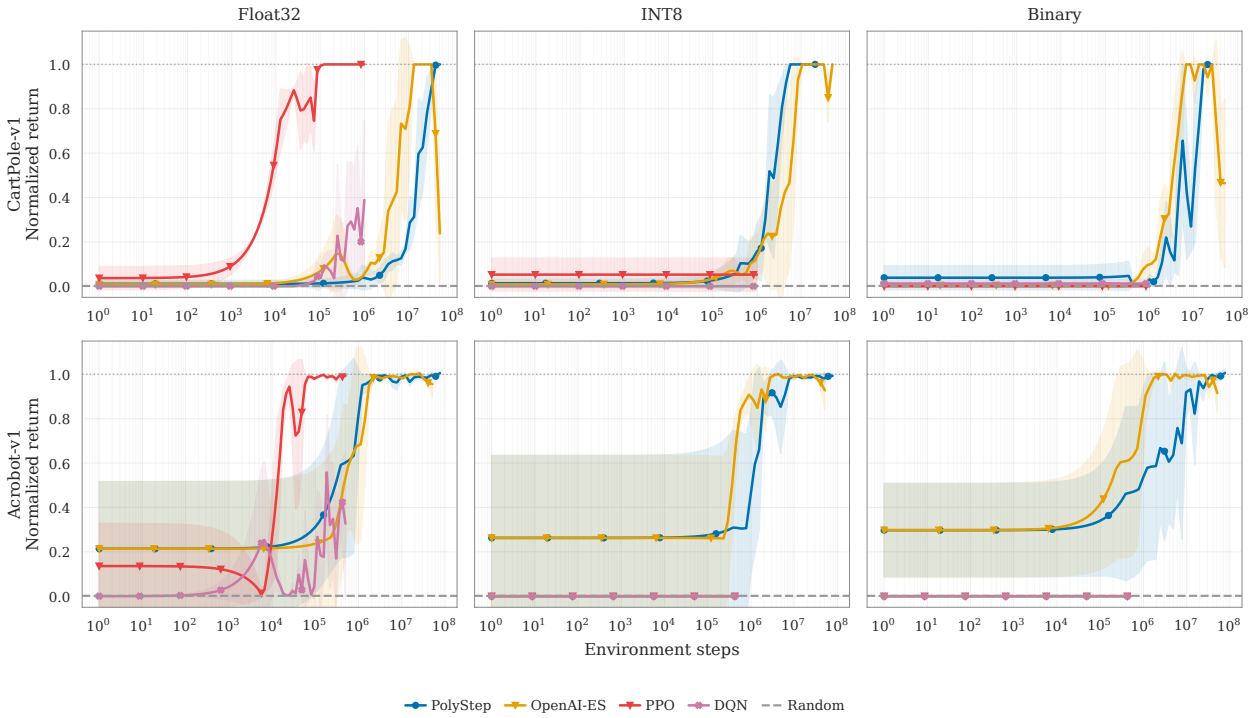

Figure 4: RL learning curves: 2 environments × 3 precision regimes (3 seeds, mean ± std). PolyStep and OpenAI-ES retain Float32 performance under INT8/Binary quantization; PPO/DQN collapse to the random floor when gradient flow is blocked.

Two limitations. OpenAI-ES is roughly 5× more sample-efficient than PolyStep to first reach a 495-return on CartPole-Float32, and reaches the −100 threshold on Acrobot-Binary ∼10× faster; asymptotic returns are nevertheless statistically indistinguishable. On larger-scale humanoid locomotion (Unitree G1, 15-DoF, ∼10K-parameter MLP policy), PPO with full-gradient access reaches a return that is ∼20× larger than PolyStep's under the same simulator and budget (Appendix E). We attribute this gap to two factors: (i) the parameter dimensionality (∼10K parameters exceeds the regime where forward-only optimization scales efficiently per Section 6.3), and (ii) the smooth differentiable simulator, where PPO's value-function machinery is genuinely advantageous. PolyStep's value appears in the quantized-policy regime where PPO collapses, not in unrestricted full-precision continuous control.

5.8 LSTM Time-Series Forecasting

To test whether gradient-free training extends beyond classification to continuous prediction tasks, we train a VmapSafeLSTM model on the ETTh1 oil temperature forecasting benchmark (Zhou et al., 2021). The model (23K parameters: LSTM hidden=64, linear head) takes a 96-step lookback window and directly predicts 96 future steps. We compare against a persistence baseline (repeat last observed value), which provides a strong floor for smooth temporal data.

All learned methods miss the persistence baseline (Table 8), reflecting the underlying signal's smoothness. Among learned methods, PolyStep attains the lowest MSE (0.121 ± 0.004), beating Adam (0.187 ± 0.026) despite seeing only forward losses. We select checkpoints by validation loss (honest protocol); the tight variance (σ=0.004) reflects val-selected stability. Config: CosineEpsilon (ϵ: 10→0.1, r_s: 5→1), wider probe (r_p: 10→2), momentum (0.5→0.95). Wider r_p is the key lever; disabling momentum collapses MSE 0.12→0.44. Among other gradient-free methods, OpenAI-ES (0.237) clears SPSA (1.279) and CMA-ES (2.126); CMA-ES's failure reflects its known difficulty above ∼10K parameters (separable CMA-ES loses cross-parameter covariance). On this regression benchmark with noisy gradient information, OT-guided polytope probing beats gradient-based optimization among learned methods. LSTM details are in Appendix D.

Table 8: ETTh1 oil-temperature forecasting (H=96, z-score normalized, 5 seeds, mean ± std). PolyStep achieves the lowest MSE among learned methods (0.121), outperforming gradient-based Adam (0.187). Persistence (non-learning baseline) sets the floor; best learned method in bold.

Method	MSE (↓)	MAE (↓)
Persistence (naive)	0.069	0.202
PolyStep	$\mathbf{0.121 \pm 0.004}$	$\mathbf{0.276 \pm 0.005}$
OpenAI-ES	0.237 ± 0.014	0.403 ± 0.012
SPSA	1.279 ± 0.095	1.078 ± 0.043
CMA-ES	2.126 ± 0.154	1.405 ± 0.114
Adam	0.187 ± 0.026	0.359 ± 0.031

5.9 Bare Ablation: Method vs. Engineering

To isolate the contribution of the core OT-guided update from the solver acceleration features (EMA amortization, biased rotation, Anderson acceleration, adaptive overrelaxation, dual momentum), we run PolyStep with *all* such features disabled. The core epsilon/radius schedules and subspace rank remain identical. Full details on the turbo features are provided in Appendix F.

Table 9: Bare ablation: PolyStep accuracy with all acceleration features disabled (5 seeds, mean ± std). Bare = no biased rotation, amortization, or Anderson acceleration. ES baselines shown for reference.

Task	**Bare PolyStep**	**Turbo PolyStep**	**OpenAI-ES**	**CMA-ES**
MNIST	$\mathbf{96.2 \pm 0.1}$	96.0 ± 0.1	87.1	61.5
SNN (LIF)	86.7 ± 8.7	$\mathbf{93.4 \pm 0.3}$	33.1	16.2

Table 9 reveals two findings. First, on MNIST, bare PolyStep (96.2%) matches turbo PolyStep (96.0%): the turbo features add *no accuracy benefit* on smooth differentiable landscapes. The entire 9-point advantage over OpenAI-ES (87.1%) and 35-point advantage over CMA-ES (61.5%) comes from the OT-guided polytope update rule itself, not from engineering acceleration. Second, on SNN, turbo features provide critical *stability*: bare PolyStep reaches 93.5–93.8% on 3 of 5 seeds but collapses to 73–79% on the other 2 (σ=8.7%), while turbo PolyStep is consistently 93.0–93.6% (σ=0.3%). The key stabilizer is biased rotation (seeding polytope rotations toward the previous descent direction), which prevents the optimizer from "forgetting" productive search directions in the chaotic spike-threshold landscape. Even in the bare configuration, PolyStep's *worst* seed (73.5%) more than doubles the best ES baseline (33.1%), confirming the method-level advantage.

Amortization sensitivity. Among the turbo features, EMA transport-plan reuse (the plan-reuse interval T_{amort}) is the most effective yet the most fragile. On MNIST's smooth landscape, reusing the transport plan for 3 consecutive steps provides 2–3× wall-clock speedup with negligible accuracy cost. However, on non-differentiable objectives, where the loss landscape changes discontinuously between steps, stale plans become catastrophic: our dose-response sweep shows that T_{amort}=10 collapses test accuracy to 24% on MNIST despite reaching 59% mid-training, a −45 pp terminal drop (Appendix F). All non-differentiable experiments in this paper therefore use T_{amort}=1 (fresh solve every step).

Limitations. A full discussion appears in Section 6.3; we summarize the key failure mode. On SST-2 (Socher et al., 2013) (4.2M-parameter transformer trained from scratch), all gradient-free methods collapse to near-random accuracy (PolyStep: 49.4%, SPSA: 53.1%, seed 42). This is not a solver limitation but a *structural* one: the rank-64 subspace captures <0.002% of the full parameter space, leaving the optimizer with an effectively random search. Modern zeroth-order methods such as MeZO (Malladi et al., 2023) achieve ∼91% on SST-2, but only by fine-tuning pretrained LLMs (OPT-1.3B+) where the effective search dimensionality is drastically reduced by the pretrained loss landscape. When we replicate this regime, freezing a pretrained GPT-2 backbone and optimizing only the 1,538-parameter classification head, PolyStep reaches 76.8% in full-space mode (Appendix P), confirming compatibility with pretrained features when the search space is tractable. The per-step compute cost is inherent to all zeroth-order methods: each PolyStep step

evaluates $2d_{\text{sub}}$ forward passes (∼20M on MNIST vs. ∼14,000 gradient steps for Adam). Scalability analysis (parameter scaling, sparse projection, `torch.compile`) is deferred to Appendix G.

5.10 Ablation Studies

We conduct ablation studies on MNIST to isolate the effect of key design choices, drawing on a comprehensive ablation sweep totalling 169 GPU-hours across 120 experimental cells (Appendix H). We focus on three dimensions: subspace mode, probe/step radius, and particle dimension. Full ablation results including polytope type, epsilon schedule, blockwise strategy, schedule fragility, and convergence analysis are provided in Appendix H.

Subspace comparison. We compare the four subspace modes defined in Table 1: full-space optimization (no projection), HybridSubspace (per-layer random projection with rank 4), LinearSubspace (global random projection with rank 4), and AdaptiveSubspace (displacement-informed global projection with rank 4096). HybridSubspace achieves the highest accuracy (93.7% ± 0.2%), followed by LinearSubspace (86.1% ± 17.2%, highly variable across seeds), full-space (84.2% ± 0.2%), and AdaptiveSubspace (71.5% ± 1.3%, best accuracy during training; some seeds exhibit significant accuracy collapse after peak, with final accuracy as low as 22%). The ±17.2% standard deviation on LinearSubspace reflects seed-level *instability* rather than mere suboptimality (worst seed degrades far below 70%); this is a sharper failure mode than the small-but-consistent gap exhibited by full-space, and we recommend HybridSubspace for any production use of PolyStep. HybridSubspace provides the best accuracy–efficiency tradeoff; its per-layer structure preserves layer-specific geometry while compressing each layer's contribution (Appendix H.1). AdaptiveSubspace's poor performance, and training instability, reflects the difficulty of global random projection with rotation at this scale: the projection changes each step, preventing the OT solver from building on previous solutions.

Radius sensitivity. The probe radius r_p and step radius r_s control the exploration–exploitation tradeoff. A probe radius of r_p=1.0 provides a robust default across tasks: smaller values underexplore, while larger values cause the cost matrix to become uninformative as probe points land in flat regions. The step radius is more task-dependent and interacts with the epsilon decay schedule.

Key findings. Four additional ablation insights emerge from our experiments: (1) Lower particle dimension yields higher accuracy in subspace mode: d_p=2 achieves 95.7% ± 0.2%, d_p=4 achieves 94.9% ± 0.2%, and d_p=8 achieves 93.6% ± 0.1% (HybridSubspace rank 4, 10 epochs, 5 seeds). With rank 4, d_p=2 creates 2 particles each exploring 4 orthoplex vertices, while d_p=8 creates a single particle with 16 vertices. More particles enable more independent local searches, outweighing the richer per-particle OT signal from higher-dimensional polytopes. (2) The orthoplex polytope (cross-polytope) provides the best accuracy among the three tested geometries (orthoplex, simplex, cube), consistent with its axis-aligned probe structure being well-suited to parameter-space exploration. (3) The epsilon decay schedule materially affects convergence: starting too low prevents early exploration, while decaying too slowly wastes function evaluations on coarse transport plans. (4) **Schedule fragility** (Proposition 4.10): cosine ε-schedules incur a 2.3 pp accuracy penalty and 4× variance amplification compared to flat schedules, independent of model size (validated at 102K and 1M parameters; Table 16 in Appendix H).

Why entropic OT? Update rule ablation. A natural question is whether the full entropic optimal transport machinery is necessary, or whether simpler vertex-weighting rules would suffice. We compare four update rules on MNIST (HybridSubspace, rank 8, 10 epochs, 5 seeds), all sharing identical cost evaluation:

- **Entropic OT** (baseline): Solve the Sinkhorn problem to obtain transport weights T_{pv}; update via barycentric projection $x_p \leftarrow \sum_v (T_{pv}/a_p)\, v_{pv}$.
- **Softmax-weighted**: $w_{pv} = \text{softmax}(-C_{pv}/\varepsilon)$; same barycentric projection but without OT marginal constraints. Each particle computes weights independently.
- **Min-cost greedy**: $x_p \leftarrow v_{p,\arg\min_v C_{pv}}$. Each particle jumps to its single lowest-cost vertex.
- **Top-k mean**: Average the k=3 lowest-cost vertices per particle uniformly.

Table 10: Update-rule ablation (MNIST, 5 seeds, mean ± std). Smooth cost-weighted interpolation (OT/softmax) is essential: both reach 96.4%; greedy and top-k collapse to ∼12%.

Update Rule	**Accuracy (%)**	**Key Property**
Entropic OT	96.4 ± 0.2	Smooth + marginal constraints
Softmax-weighted	96.4 ± 0.2	Smooth, no constraints
Min-cost greedy	12.8 ± 0.8	Hard selection
Top-k mean	11.7 ± 0.5	Uniform over 3 best

Table 11: Full-space OT vs. softmax (MNIST, P=50,885 particles, V=4 vertices, 5 seeds). The OT target-marginal constraint yields a +23 pp advantage when many particles compete for few vertices.

Update Rule	**Subspace (few P)**	**Full-space (P=50K)**
Entropic OT	96.4 ± 0.2	57.6 ± 11.0
Softmax-weighted	96.4 ± 0.2	34.6 ± 13.9
Gap	0.0	**+23.0**

The results (Table 10) reveal that *smooth cost-weighted interpolation is the essential mechanism*: both entropic OT and softmax achieve 96.4%, while greedy selection and uniform top-k averaging collapse to random chance (∼12%). The failure of greedy and top-k rules is instructive: jumping to a single vertex (greedy) or uniformly averaging a few best vertices (top-k) produces noisy, high-variance updates that prevent convergence. The smooth weighting provided by both OT and softmax acts as implicit regularization, producing stable descent directions by blending information from all vertices proportional to their cost.

Theoretical relationship. Entropic OT and softmax are connected: when the OT problem has a single source particle, the solution reduces exactly to $\text{softmax}(-C/\varepsilon)$ (one-sided entropic OT (Litman, 2025)). In subspace mode with HybridSubspace (rank 8, d_p=8), the OT problem has relatively few particles ($P = d_{\text{sub}}/8$) and 16 vertices. With few particles and diverse cost profiles, the target marginal constraint is naturally satisfied and the Sinkhorn dual potential $g_v \approx 0$, so each row of the transport plan converges to independent softmax, explaining the empirical equivalence.

The OT formulation provides theoretical advantages when *many particles compete for few vertices with correlated cost profiles*: the target marginal $\sum_p T_{pv} = b_v$ prevents all particles from collapsing toward the same low-cost vertex, and the dual potential g_v acts as a price signal that redirects marginal particles toward undersubscribed vertices. We verify this empirically by running both methods in full-space mode on MNIST (P=50,885 particles, V=4 vertices, 3 epochs, 5 seeds):

Table 11 confirms the theoretical prediction: when many particles compete for few vertices, the target marginal $\sum_p T_{pv} = b_v$ prevents collapse to a single search direction, yielding a 23-point accuracy advantage over independent softmax weighting. The high variance in full-space mode reflects the challenging optimization landscape (101K parameters, no subspace compression), but entropic OT consistently outperforms softmax across all seeds.

Regime map: softmax default, OT as principled fallback. The combined evidence from Tables 10 and 11 forms a two-axis *regime map* for the OT-vs-softmax choice (Figure 5 (c)): plotting accuracy gap (OT − softmax) against subspace rank yields a smooth transition from 0 pp at small particle-to-vertex ratios to +23 pp at $P/V \approx 12{,}721$ (full-space MNIST with P=50,885 particles competing for V=4 vertices). In the subspace regime, which produces every headline number in this paper, softmax is the practical default: a single fused kernel per step, no iterations, no dual potentials, mathematically equivalent to entropic OT (both at 96.4%). A dedicated rank sweep (ranks 2–32, 3 seeds each) confirms that this equivalence holds across the entire measured rank range: softmax and Sinkhorn are statistically indistinguishable at every rank from 2 to 16, with no particle-to-vertex crossover threshold observed in the subspace mode. In the high-particle regime where the target marginal becomes binding (full-space mode, $P \gg V$), entropic OT is the principled generalization that recovers the diversity guarantee softmax loses; the +23 point gap is the price of dropping the target constraint. The regime map therefore positions softmax as the right operational

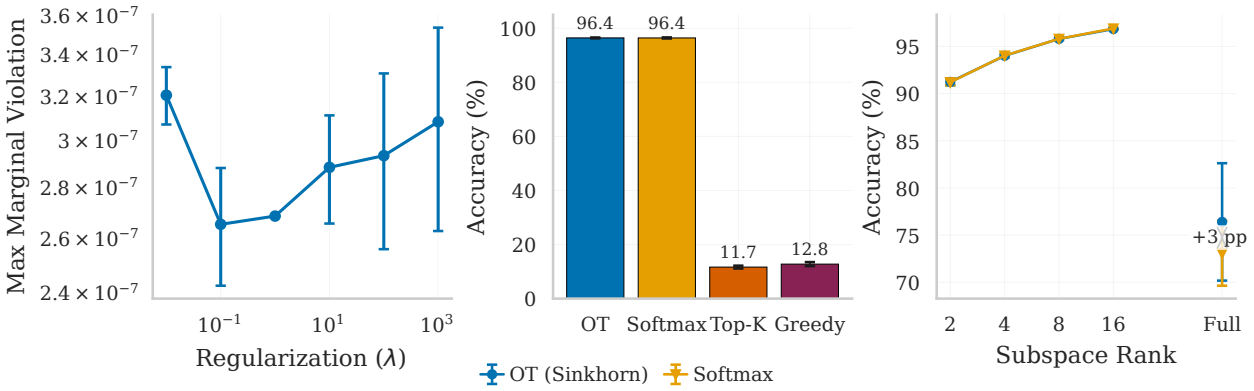

Figure 5: OT solver analysis. (a) KL marginal violation decreases monotonically with λ (empirical anchor for Theorem 4.1; fitted log-log slope $\alpha \approx -0.03$, indicating λ acts mainly as a regularization knob in the operating regime). (b) Update-rule accuracy: smooth weighting essential (Table 10). (c) OT vs. softmax: equivalent across subspace ranks 2–16; OT gains +23 pp in full-space (P=50K).

default and entropic OT as the theoretically sound extension that comes into its own precisely when softmax breaks down.

6 Discussion

6.1 Summary of Contributions

Softmax-weighted polytope probing guides gradient-free updates for neural networks at practical scale, especially on *non-differentiable* models. Entropic OT is the *analytical lens* that predicts when softmax suffices: in the common subspace regime the column marginal is non-binding and softmax recovers OT exactly (Theorem 4.1); when many particles compete for few vertices, OT yields a +23 pp gain on full-space MNIST (Table 11). Four empirical findings: (1) training of genuinely non-differentiable architectures (93.4% SNN, 97.2% INT8, 86.8% argmax attention, 93.2% staircase, 90.7% hard MoE) without surrogates or STE; (2) MAX-SAT satisfaction >92% from 100 to 10^6 variables; (3) 29.8× memory reduction over BPTT at T=400; (4) 96.0% on MNIST and 0.121 MSE on ETTh1.

6.2 When to Use PolyStep

PolyStep fills a specific niche: models where gradients do not exist and no workaround (STE, surrogate gradients, relaxation) is satisfactory. Concrete use cases include:

- **Hard non-differentiable components**: LIF spiking neurons, integer quantization (rounding), argmax attention, discrete routing, where STE introduces bias and DeepZero/MeZO produce zero gradient estimates.
- **Black-box forward passes**: external simulators, hardware-in-the-loop, or proprietary modules where only the loss scalar is observable.
- **RL policy search**: deterministic or non-differentiable policies evaluated through simulators, including high-throughput MJWarp robotics loops where batched rollouts make scalar-return optimization practical.
- **Non-differentiable losses**: Hamming distance, edit distance, combinatorial objectives where the loss function itself has zero gradient.

PolyStep should *not* be used when gradients are available: Adam with backpropagation is faster, more accurate, and scales better. For differentiable networks at scale, DeepZero (Chen et al., 2024) and MeZO (Malladi et al., 2023) are superior zeroth-order alternatives. PolyStep's value proposition is strictly for the non-differentiable regime.

6.3 Limitations

Despite the memory advantage for SNN evaluation, PolyStep has clear limitations.

Positioning vs. prior gradient-free methods. We position PolyStep against three families.

Evolution strategies. CMA-ES (Hansen, 2016) and OpenAI-ES (Salimans et al., 2017) are well-established baselines. The distinction is *algorithmic*: OpenAI-ES estimates a smoothed gradient $\hat{g} = \frac{1}{N\sigma}\sum_i \mathcal{L}(\theta+\sigma\epsilon_i)\,\epsilon_i$ ($\epsilon_i \sim \mathcal{N}(0, I)$). PolyStep instead solves the OT problem (Eq. 2) over a cost matrix $\boldsymbol{C} \in \mathbb{R}^{P\times V}$ at structured polytope vertices and applies the barycentric step (Eq. 6). The update is a nonlinear soft-argmin over the vertex set, exploiting the full cost matrix jointly rather than using perturbations as independent noisy gradient components.

Zeroth-order for deep differentiable networks. DeepZero (Chen et al., 2024) reaches ∼90% on CIFAR-10 (ResNet-20) via coordinate-wise ZO-SGD; MeZO (Malladi et al., 2023) fine-tunes LLMs at inference-level memory. Both assume differentiable forward passes, so coordinate perturbations through hard thresholds (rounding, sign, argmax) produce zero estimates. We do not compete with them on differentiable networks; they are superior there.

Forward-only training. Forward-Forward (Hinton, 2022) attains ∼99% on MNIST via layer-wise contrastive learning; NoProp (Li et al., 2025) matches backprop on CIFAR-10 via block-wise denoising. Both require architectural modifications (goodness functions, denoising heads) and NoProp still uses local within-block gradients. We train most standard `nn.Module`s with VmapSafe drop-ins for attention and recurrent layers, and use no gradient at any level.

Where the contribution sits. The contribution of PolyStep is not gradient-free optimization of differentiable networks, which is well-solved. It is training models with *genuinely non-differentiable* forward passes (hard LIF thresholds, integer rounding, argmax attention, discrete MoE routing) where DeepZero/MeZO produce zero gradients, Forward-Forward requires incompatible architectures, and STE introduces approximation bias. PolyStep's polytope-probing approach does not estimate gradients; it evaluates the loss at structured polytope vertices and applies a soft-assignment update (Eq. 1, with entropic OT (Eq. 2) as a strict generalization), making it applicable to any forward-evaluable function regardless of differentiability. Concretely, the update rule (Eq. 6) computes a Wasserstein-projected displacement on the polytope geometry: the soft-assignment plan $\boldsymbol{T}^*$ acts as a *data-dependent, cost-aware weighting* over candidate directions, contrasting with ES methods that use random perturbations as basis vectors for gradient regression. This mechanism explains PolyStep's advantage on non-differentiable objectives: it requires only that cost differences between vertices be informative, not that the loss be smooth or admit finite-difference estimates. The OT formulation provides the theoretical underpinning, establishing the soft assignment as a regularized optimal transport plan that recovers softmax in the common subspace regime and adds a target-marginal diversity guarantee in the high-particle regime (Section 5.10), while the softmax solver is the practical algorithm used to produce every reported number. To our knowledge, no prior work has systematically evaluated gradient-free optimization on such architectures with 5-seed statistical validation.

Accuracy gap on differentiable tasks. On tasks where exact gradients are available, gradient-based methods retain a significant accuracy advantage (Section 5.6). PolyStep is not a replacement for backpropagation on differentiable networks, and should not be used when gradients are accessible.

Non-differentiable model success and limits. PolyStep successfully trains a spiking neural network with hard LIF thresholds to 93.4% accuracy (Section 5.3), while other gradient-free baselines achieve only 16–33% on the same task despite receiving comparable compute budgets (2000 generations for ES, 10000 iterations for SPSA). However, gradient-free methods remain limited on tasks where small parameter perturbations do not produce informative cost differences, a challenge expected to be more severe on sparse event-driven data (e.g., neuromorphic spike streams) where outputs are discrete and high-dimensional.

High-dimensional NLP. At 4.2M parameters (SST-2 transformer trained from scratch), all gradient-free methods collapse to near-random accuracy (PolyStep: 49.4%, SPSA: 53.1%, single seed) using a rank-

64 subspace that covers <0.002% of the parameter space. This empirical scale ceiling is consistent with theoretical lower bounds on zeroth-order optimization: Duchi et al. (2015) and Jamieson et al. (2012) establish that ZO methods incur a $\sqrt{d}$ query-complexity penalty over first-order methods even on smooth convex objectives, and the gap is strictly worse on non-smooth objectives. For $d = 4.2$M, this penalty exceeds 2,000×, making from-scratch training of full-scale transformers infeasible for any forward-only method, including DeepZero, MeZO, and PolyStep, unless the parameter space is first compressed by a pretrained landscape. The failure is therefore not the OT solver that breaks but the *subspace coverage*: the random projection discards too much of the language-model geometry to produce informative cost differences between probe points, exactly as the lower bounds predict. This contrasts sharply with zeroth-order methods that fine-tune pretrained LLMs (e.g., MeZO (Malladi et al., 2023) achieves ∼91% on SST-2 with OPT-1.3B), where the pretrained loss landscape concentrates the effective search dimensionality into a low-rank manifold that ZO perturbations can navigate. When we replicate this regime, freezing a pretrained GPT-2 backbone and optimizing only the 1,538-parameter classification head, PolyStep reaches 76.8% (Appendix P), confirming that the OT-guided update is compatible with pretrained features once the search dimensionality is tractable. The head-only fine-tuning regime is the practical operating point for forward-only methods at NLP scale.

Per-step compute cost. Each step uses $2K \cdot d_{\text{sub}}$ forward passes (orthoplex case of the general $P \times V \times K$ count from Section 3.1). On MNIST (K=1), this is ∼20M model forwards over 3,540 steps, vs. ∼14K forward-backward passes for Adam, a three-order-of-magnitude evaluation gap. Adam's backward pass yields d-dimensional gradient information; each of our forward passes yields only a scalar. This is inherent to ZO methods. DeepZero (Chen et al., 2024) mitigates via coordinate-wise estimation but requires differentiability; we pay the full ZO cost in exchange for non-differentiable applicability. Table 12 (single RTX 5090) confirms 10–100× slowdowns vs. Adam, while we remain comparable to CMA-ES/OpenAI-ES on most tasks. MoE is the apparent outlier (∼30× slower than OpenAI-ES), but is also where we lead by 25+ pp (90.7% vs. 62–69%) in a regime Adam cannot enter (the "–" entry). The wall-clock there is the entry price for a training regime no gradient-based method can reach.

Table 12: Mean wall-clock time (seconds) per method and benchmark, single RTX 5090 GPU, 5 seeds. PolyStep is consistently slower than Adam but competitive with other gradient-free methods.

Benchmark	**PolyStep**	**CMA-ES**	**OpenAI-ES**	**SPSA**	**Adam**
MNIST	341	162	116	21	9
SNN (LIF)	3,769	1,896	900	213	23
INT8	1,500	1,787	409	100	8
Staircase	988	2,402	342	95	10
Argmax	2,741	2,974	908	1,450	81
MoE	9,844	325	244	468	–
ETTh1	758	669	764	125	151

RL sample efficiency. The RL experiments should be read as direct policy-search evidence, not as a claim that PolyStep replaces PPO. RSL-RL PPO on Unitree G1 uses trajectory structure, a value function, advantage estimation, and many gradient updates per rollout batch. PolyStep sees only scalar returns for candidate policies. This makes it naturally compatible with hard action selection, quantized policies, proprietary simulators, and hardware-in-the-loop evaluation, but it also means PPO remains the sample-efficiency reference whenever gradients and standard policy-gradient assumptions are available.

Vectorized evaluation bottleneck. For MLP-only models we bypass `vmap` with explicit `torch.bmm`, mapping N parameter configurations into the hardware batch dimension (a single fused kernel per layer). For attention, convolution, and normalization layers, no fused path exists; the general `vmap` path pays per-op dispatch and materializes all N intermediate activations simultaneously. In our profiling, `vmap` is ∼3× slower than the batched-linear fast path at identical parameter counts. Closing this gap needs architecture-specific batched evaluators or mature `torch.compile`-on-`vmap`, both future work (Appendix L).

6.4 Future Directions

Several extensions look promising. Pretrained embeddings for NLP would shrink the effective search dimension; our head-only GPT-2 result (76.8% on SST-2, 1,538 trainable parameters, Appendix P) is initial evidence. Richer subspaces (per-layer adaptive projections, learned bases) could enlarge the trainable fraction. Hybrid recipes that apply OT-guided updates only to non-differentiable components while backpropagating elsewhere are a natural step. Scaling to larger vision (CIFAR-10 with deeper backbones), longer SNN horizons ($T > 1000$), and systematic scaling laws over subspace rank are deferred to future work.

Hardware roadmaps increasingly favor inference over training throughput: inference-only accelerators are entering production. Each of our evaluations is a plain forward pass with no gradient tape, no activation checkpointing, and no optimizer state. Distributed execution also differs: BPTT needs AllReduce of $O(D)$ gradients, while forward-only training broadcasts parameters and gathers $O(n_{\text{probes}})$ scalars (Appendix O). This is a forward-looking observation, not a current advantage. We still need orders of magnitude more forward evaluations than Adam needs gradient steps ($\sim$20M vs. $\sim$14K on MNIST). As inference scales faster than training, this gap may narrow.

7 Conclusion

We have presented PolyStep, a gradient-free neural network optimizer that updates parameters via softmax-weighted polytope probing and barycentric projection, requiring only forward passes. The softmax update is the one-sided limit of a regularized optimal-transport problem over the same polytope geometry, which serves as the principled extension when many particles compete for few vertices.

The central contribution is training neural networks with genuinely non-differentiable forward passes (spiking neurons, quantized layers, argmax routing, staircase activations, and hard MoE gating), a setting where existing gradient-free methods (DeepZero, MeZO, Forward-Forward) cannot operate because they require differentiability or architectural compatibility. Beyond supervised learning, PolyStep scales gracefully on combinatorial objectives at million-variable MAX-SAT and matches OpenAI-ES on RL policy search while retaining performance under hard policy quantization.

PolyStep fills a specific gap: when the model contains components with zero or undefined gradients and no surrogate is satisfactory, OT-guided polytope probing provides a principled training signal from forward evaluation alone. On differentiable benchmarks Adam remains faster and more accurate, a limitation shared by all zeroth-order methods and consistent with the $\sqrt{d}$ query-complexity lower bounds (Theorem 4.3). Each step reduces to a batched forward pass, aligning with inference-optimized hardware trajectories; narrowing the evaluation gap ($\sim$20M forward passes vs. $\sim$14K Adam steps on MNIST) is a prerequisite for forward-only training to become broadly competitive.

The polytope-OT primitive, originally introduced for motion planning by Le et al. (2023), now scales to neural-network training via the subspace and non-differentiable extensions presented here, suggesting a generally applicable optimization tool for forward-evaluable objectives rather than a domain-specific recipe.

Code is available at `https://anonymous.4open.science/r/polystep`.

References

Youhei Akimoto and Nikolaus Hansen. Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary Computation*, 28(3):405–435, 2020.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burber, et al. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, volume 2, pp. 929–947, 2024.

Adrian Balint and Uwe Schöning. Choosing probability distributions for stochastic local search and the role of make versus break. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pp. 16–29. Springer, 2013.

Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Jérôme Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

James V. Burke, Frank E. Curtis, Adrian S. Lewis, Michael L. Overton, and Lucas E. A. Simões. Gradient sampling methods for nonsmooth optimization. *Numerical Nonsmooth Optimization (Springer)*, 2020.

Shaowei Cai, Zhendong Lei, and Xindi Zhang. Local search for SAT: A review of recent advances. *Information Sciences*, 547:1024–1041, 2021.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. DeepZero: Scaling up zeroth-order optimization for deep model training. In *International Conference on Learning Representations (ICLR)*, 2024.

Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

Frank H. Clarke. *Optimization and Nonsmooth Analysis*, volume 5 of *Classics in Applied Mathematics*. SIAM, 1990.

Noa Cohen, Omkar Joglekar, Dotan Di Castro, Vladimir Tchuiev, Shir Kozlovsky, and Michal Moshkovitz. Gradient-free training of quantized neural networks. *arXiv preprint arXiv:2410.09734*, 2024.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Sander Dalm, Marcel van Gerven, and Nasir Ahmad. Node perturbation can effectively train multi-layer neural networks. *arXiv preprint arXiv:2310.00965*, 2023.

Giorgia Dellaferrera and Gabriel Kreiman. Error-driven input modulation: Solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning (ICML)*, 2022.

John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 385–394, 2005.

Google DeepMind and NVIDIA. MuJoCo Warp: Gpu-optimized mujoco simulation with nvidia warp. `https://github.com/google-deepmind/mujoco_warp`, 2025.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Morris W. Hirsch. *Differential Topology*, volume 33 of *Graduate Texts in Mathematics*. Springer, 1976.

Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. RC2: An efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11:53–64, 2019.

Kevin G. Jamieson, Robert Nowak, and Benjamin Recht. Query complexity of derivative-free optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Halyun Jeong, Jack Xin, and Penghang Yin. Beyond discreteness: Finite-sample analysis of straight-through estimator for quantization. *arXiv preprint arXiv:2505.18113*, 2025.

David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

An T. Le, Georgia Chalvatzaki, Armin Biess, and Jan R. Peters. Accelerating motion planning via optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 287–296. ACM, 2006.

Qinyu Li, Yee Whye Teh, and Razvan Pascanu. NoProp: Training neural networks without full back-propagation or full forward-propagation. *arXiv preprint arXiv:2503.24322*, 2025.

Elon Litman. Scaled-dot-product attention as one-sided entropic optimal transport. *arXiv preprint arXiv:2508.08369*, 2025.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. Sparse MeZO: Less parameters for better performance in zeroth-order LLM fine-tuning. *arXiv preprint arXiv:2402.15751*, 2024.

Ilya Loshchilov. LM-CMA: An alternative to L-BFGS for large-scale black-box optimization. *Evolutionary Computation*, 25(1):143–171, 2017.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with Gumbel-Sinkhorn networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory- and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6166–6176, 2023.

Francesco Mezzadri. How to generate random matrices from the classical compact groups. *Notices of the AMS*, 54(5):592–604, 2007.

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019.

Alberto Pirillo, Luca Colombo, and Manuel Roveri. NITRO-D: Native integer-only training of deep convolutional neural networks. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*, 2025. Preprint arXiv:2407.11698.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Herbert Robbins and David Siegmund. A convergence theorem for non-negative almost supermartingales and some applications. In *Optimizing Methods in Statistics*, pp. 233–257. Academic Press, 1971.

R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Michael E. Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Meyer Scetbon, Marco Cuturi, and Gabriel Peyré. Low-rank Sinkhorn factorization. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Sholom Schechtman. The gradient's limit of a definable family of functions admits a variational stratification. *SIAM Journal on Optimization*, 2024. To appear; preprint arXiv:2402.08272.

Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations (ICLR)*, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013.

James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable ADMM approach. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 2722–2731, 2016.

Mauricio Ortiz Torres, Markus Lange, and Arne P. Raulf. On advancements of the forward-forward algorithm. *arXiv preprint arXiv:2504.21662*, 2025.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15:949–980, 2014.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Changqing Xu, Ziqiang Yang, Yi Liu, Xinfang Liao, Guiqi Mo, Hao Zeng, and Yintang Yang. FFGAF-SNN: The forward-forward based gradient approximation free training framework for spiking neural networks. *arXiv preprint arXiv:2507.23643*, 2025.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations (ICLR)*, 2019.

Ziming Yu, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of LLMs in random subspaces. *arXiv preprint arXiv:2410.08989*, 2024.

Pengyun Yue, Xuanlin Yang, Mingqing Xiao, and Zhouchen Lin. PseuZO: Pseudo zeroth-order algorithm for training deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kevin Zakka, Qiayuan Liao, Brent Yi, Louis Le Lay, Koushil Sreenath, and Pieter Abbeel. mjlab: A lightweight framework for gpu-accelerated robot learning. arXiv:2601.22074, 2026.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*, 2021.

A Proofs

We collect the full proofs for the theorems and corollaries stated in Section 4. Notation follows that section: $\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{P\times V}$ is a transport plan, $\boldsymbol{C} \in \mathbb{R}^{P\times V}$ a cost matrix, $\boldsymbol{a} \in \Delta_P$ the source marginal (uniform), $\boldsymbol{b} \in \Delta_V$ the target marginal (uniform), $\varepsilon > 0$ the entropic temperature, and $\lambda \in [0, +\infty]$ the KL-penalty strength. We write $H(\boldsymbol{P}) = -\sum_{i,j} P_{ij} \log P_{ij}$ for the negative entropy and $D_{\mathrm{KL}}(\boldsymbol{p}\|\boldsymbol{q}) = \sum_j p_j \log(p_j/q_j)$ for the Kullback-Leibler divergence (with $0 \log 0 = 0$).

A.1 Proof of Theorem 4.1

(1) Existence, uniqueness, continuity. The objective in Eq. 7 is the sum of a linear term $\langle \boldsymbol{C}, \boldsymbol{P}\rangle$, a strictly convex term $\varepsilon H(\boldsymbol{P})$ on the simplex $\{\boldsymbol{P} \geq 0, \boldsymbol{P}\mathbf{1} = \boldsymbol{a}\}$, and a non-negative convex term $\lambda D_{\mathrm{KL}}(\boldsymbol{P}^\top \mathbf{1}\|\boldsymbol{b})$ for $\lambda \geq 0$. The feasible set is non-empty (take $P^0_{ij} = a_i b_j$), compact in $\mathbb{R}^{PV}$, and the objective is lower semi-continuous and coercive on this set. By Weierstrass it admits a minimum; by strict convexity in $\boldsymbol{P}$ (recall $\varepsilon > 0$) the minimum is unique. Continuity of $\lambda \mapsto \boldsymbol{P}^\star_\lambda$ on $(0, +\infty)$ follows from Rockafellar (1970, Theorem 10.1) applied to the parametric convex program.

(2) KKT and dual potentials. We assume the marginals $\boldsymbol{a}, \boldsymbol{b}$ have full support ($a_i, b_j > 0$ for all i, j); strict positivity ensures $\log b_j$ and the entropic dual potentials are finite, and is preserved by $\boldsymbol{P}^\star_\lambda$ for $\lambda < \infty$ via interior-point optimality of the entropic objective (so $\log q^\star_j$ is also well-defined throughout). Introducing $\boldsymbol{f} \in \mathbb{R}^P$ for the row-marginal constraint $\boldsymbol{P}\mathbf{1} = \boldsymbol{a}$, the Lagrangian is

$$\mathcal{L}(\boldsymbol{P}, \boldsymbol{f}) = \langle \boldsymbol{C}, \boldsymbol{P}\rangle + \varepsilon H(\boldsymbol{P}) + \lambda D_{\mathrm{KL}}(\boldsymbol{P}^\top \mathbf{1}\|\boldsymbol{b}) - \langle \boldsymbol{f}, \boldsymbol{P}\mathbf{1} - \boldsymbol{a}\rangle.$$

Stationarity $\partial\mathcal{L}/\partial P_{ij} = 0$ at the optimum gives, after rearrangement,

$$\log P^\star_{ij} = \frac{f_i + g^\star_j - C_{ij}}{\varepsilon} - 1, \quad g^\star_j = \frac{\lambda}{\lambda + \varepsilon}\left(\log b_j - \log q^\star_j\right), \tag{13}$$

where $q^\star_j = \sum_i P^\star_{ij}$ and the α-scaling $\alpha := \lambda/(\lambda + \varepsilon)$ is the unbalanced-OT coefficient of Chizat et al. (2018, Section 3).

(3) Endpoint identification. At $\lambda = 0$, $\alpha = 0$, $g^\star_j = 0$, and substituting into Eq. 13 gives the closed-form softmax rule $P^\star_{ij} = a_i\, e^{(f_i - C_{ij})/\varepsilon - 1}$ which row-normalizes to $\mathrm{softmax}(-C_{i:}/\varepsilon)$ scaled by a_i, exactly Eq. 1.

At $\lambda \to +\infty$, $\alpha \to 1$, and Eq. 13 reduces to the standard balanced-OT KKT system: $g^\star_j = \log b_j - \log q^\star_j$, equivalently $q^\star_j = b_j\, e^{-g^\star_j}$ which combined with the row constraint enforces both marginals exactly, the entropic OT solution $\boldsymbol{P}^\star_{\mathrm{Sink}}$ of Eq. 2.

(4) Marginal-violation monotonicity. We prove that $\lambda \mapsto D_{\mathrm{KL}}(\boldsymbol{q}_\lambda \,\|\, \boldsymbol{b})$ is non-increasing on $[0, +\infty]$. Fix $0 \leq \lambda_1 \leq \lambda_2 \leq +\infty$. By optimality of $\boldsymbol{P}^\star_{\lambda_2}$ for the λ_2-objective and feasibility of $\boldsymbol{P}^\star_{\lambda_1}$ (the row marginal $\boldsymbol{P}\mathbf{1} = \boldsymbol{a}$ is common to both objectives),

$$\langle \boldsymbol{C}, \boldsymbol{P}^\star_{\lambda_2}\rangle + \varepsilon H(\boldsymbol{P}^\star_{\lambda_2}) + \lambda_2 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2}\|\boldsymbol{b}) \;\leq\; \langle \boldsymbol{C}, \boldsymbol{P}^\star_{\lambda_1}\rangle + \varepsilon H(\boldsymbol{P}^\star_{\lambda_1}) + \lambda_2 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1}\|\boldsymbol{b}). \tag{14}$$

By optimality of $\boldsymbol{P}^\star_{\lambda_1}$ for the λ_1-objective and feasibility of $\boldsymbol{P}^\star_{\lambda_2}$,

$$\langle \boldsymbol{C}, \boldsymbol{P}^\star_{\lambda_1}\rangle + \varepsilon H(\boldsymbol{P}^\star_{\lambda_1}) + \lambda_1 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1}\|\boldsymbol{b}) \;\leq\; \langle \boldsymbol{C}, \boldsymbol{P}^\star_{\lambda_2}\rangle + \varepsilon H(\boldsymbol{P}^\star_{\lambda_2}) + \lambda_1 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2}\|\boldsymbol{b}). \tag{15}$$

Adding Eq. 14 and Eq. 15 and canceling the common $\langle \boldsymbol{C}, \cdot\rangle + \varepsilon H(\cdot)$ terms,

$$\lambda_2 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2}\|\boldsymbol{b}) + \lambda_1 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1}\|\boldsymbol{b}) \;\leq\; \lambda_2 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1}\|\boldsymbol{b}) + \lambda_1 D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2}\|\boldsymbol{b}),$$

which rearranges to

$$(\lambda_2 - \lambda_1)\left[D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1}\|\boldsymbol{b}) - D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2}\|\boldsymbol{b})\right] \;\geq\; 0.$$

For $\lambda_2 > \lambda_1$ this gives $D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_2}\|\boldsymbol{b}) \leq D_{\mathrm{KL}}(\boldsymbol{q}_{\lambda_1}\|\boldsymbol{b})$, the monotonicity statement of Eq. 8. The case $\lambda_2 = \lambda_1$ is trivial.

The vanishing limit $D_{\mathrm{KL}}(\boldsymbol{q}_\lambda \| \boldsymbol{b}) \to 0$ as $\lambda \to +\infty$ is immediate from the full-Sinkhorn endpoint identification of part (3): $\boldsymbol{P}^\star_\lambda \to \boldsymbol{P}^\star_{\mathrm{Sink}}$, whose column marginal exactly equals $\boldsymbol{b}$, and continuity of the KL divergence in the (always non-negative) plan entries gives $D_{\mathrm{KL}}(\boldsymbol{q}_\lambda \| \boldsymbol{b}) \to D_{\mathrm{KL}}(\boldsymbol{b} \| \boldsymbol{b}) = 0$.

Both statements are direct specializations of the unbalanced-OT monotonicity calculus of Chizat et al. (2018), restricted to the one-sided setting where the row penalty is infinite (hard row marginal) and the column penalty equals λ. □

A.2 Proof of Theorem 4.3

We extend the smooth-objective descent argument of Le et al. (2023, Theorem 3.2) via the Clarke generalized gradient calculus.

Step 1: Almost-sure smoothness on the probe set (transversality with radius jitter). Fix $\theta \in \mathbb{R}^d$. The probe locations $\{p_{i,v,k}\}$ from Eq. 4 depend continuously on the rotation $R_i \in \mathrm{SO}(d_p)$ *and* the radius jitter $\eta_t \in [-\eta_{\max}, \eta_{\max}]$ from Theorem 4.3 condition (iv). Random rotation alone, restricted to a finite vertex set, only randomizes within a (d_p-1)-dimensional sphere, which is a strictly lower-dimensional surface inside $\mathbb{R}^d$ when probes live in a particle subspace of dimension d_p, and a measure-zero $\mathcal{D} \subset \mathbb{R}^d$ can in principle still contain part of that surface. The radius jitter $\eta_t \sim \mathrm{Uniform}[-\eta_{\max}, \eta_{\max}]$ adds the missing radial degree of freedom: the joint (R, η)-parametrized probe map $(R, \eta) \mapsto \theta + r_p(1+\eta)\varepsilon\, R\, v_k$ is a smooth submersion onto a positive-Lebesgue-measure tube around the (d_p-1)-sphere of radius $r_p\varepsilon$, so the joint distribution of probe locations is absolutely continuous on that tube. Applying Fubini's theorem to the joint (R, η) measure (Hirsch, 1976, Ch. 3): $\mathcal{D}$ has Lebesgue measure zero in $\mathbb{R}^d$, the tube has positive measure, hence the set of (R, η) for which any probe lands on $\mathcal{D}$ has measure zero under the joint distribution. With probability one all probes lie in $\mathbb{R}^d \setminus \mathcal{D}$, $\mathcal{L}$ is C^1 at every probe, and Eq. 5 yields finite cost values.

Step 2: Surrogate gradient inclusion in the conservative field. Define the smoothed surrogate $\mathcal{L}_\varepsilon(\theta) := \mathbb{E}_{R,\eta,v}[\mathcal{L}(\theta + r_p(1+\eta)\varepsilon R\, v)]$ where the expectation is over the random rotation R, the radius jitter η, and the uniform choice of polytope vertex $v \in \mathcal{V}$. Because $\mathcal{L}$ is bounded and the joint (R, η) kernel has positive density on the tube of probe locations, $\mathcal{L}_\varepsilon$ is globally Lipschitz with constant $L_\varepsilon = O\big(\|\mathcal{L}\|_\infty / (r_p\varepsilon)\big)$ and C^∞ in θ for $\varepsilon > 0$ *regardless of whether $\mathcal{L}$ is continuous*: pointwise discontinuity of $\mathcal{L}$ across $\mathcal{D}$ is averaged out by the jitter-induced positive-measure tube. Locally, on each connected component of $\mathbb{R}^d \setminus \mathcal{D}$ where $\mathcal{L}$ is C^1, the standard spherical convolution argument applies. By Stein's identity for spherical convolutions (Flaxman et al., 2005, Lem. 2.1), $\nabla \mathcal{L}_\varepsilon(\theta) = (1/r_p\varepsilon)\, \mathbb{E}_{R,v}[v\, \mathcal{L}(\theta + r_p\varepsilon R v)]$. For locally Lipschitz $\mathcal{L}$ with measure-zero non-differentiability set, Bolte & Pauwels (2020) show that $\nabla \mathcal{L}_\varepsilon(\theta) \to g$ for some g in the *conservative gradient* of $\mathcal{L}$ at θ as $\varepsilon \to 0$, with the convergence uniform on compact sets. We do not claim convergence to a specific element of the Clarke subdifferential; Burke et al. (2020) note that Gaussian-smoothed gradients need not exhaust $\partial^\circ \mathcal{L}$ for non-Clarke-regular integrands. For *definable* losses (the four headline showcases), Schechtman (2024) shows that the gradient limit of a smoothed family is a conservative set-valued field admitting a variational stratification, which strengthens the conservative-gradient notion of Bolte & Pauwels (2020) to a regime well-aligned with the Clarke calculus.

Step 3: Stein-gradient identity for the softmax barycentric step. The softmax temperature in Algorithm 1 is the same symbol ε that scales the probe radius. The two roles must be unpacked before any Taylor argument applies, because the leading softmax-temperature term $-\mathcal{L}(\theta)/\varepsilon$ is common across all vertices and *cancels* in the softmax (which is shift-invariant in its logits). After cancellation, the residual softmax argument depends only on the centered cost $\tilde{C}_{i,v}$, which is itself $O(r_p\varepsilon)$ by Lipschitz continuity of $\mathcal{L}_\varepsilon$. The effective small parameter is therefore r_p once ε is fixed (or, equivalently, the per-step displacement $r_p\varepsilon \cdot \|\nabla \mathcal{L}_\varepsilon\|$). Write $\tilde{C}_{i,v} := C_{i,v} - \bar{C}_i$, $\bar{C}_i := V^{-1}\sum_v C_{i,v}$: shift-invariance gives $\mathrm{softmax}(-C_{i:}/\varepsilon)_v = \mathrm{softmax}(-\tilde{C}_{i:}/\varepsilon)_v$, and a first-order Taylor expansion of $\mathcal{L}$ around θ_t along the probe direction gives

$$\tilde{C}_{i,v} = r_p\, \varepsilon\, \langle \nabla \mathcal{L}_\varepsilon(\theta_t),\, R\, v_v - V^{-1} \sum_{v'} R\, v_{v'} \rangle \;+\; O\big((r_p\varepsilon)^2\big). \tag{16}$$

For the orthoplex $\{\pm e_j\}$ the centering sum vanishes ($V^{-1}\sum_{v'} v_{v'} = 0$), so $\tilde{C}_{i,v}/\varepsilon = r_p\, \langle \nabla \mathcal{L}_\varepsilon, R v_v \rangle + O(r_p^2\varepsilon)$, which is $O(r_p)$ uniformly in ε. The relevant linearization is therefore in r_p (small step), *not* in $1/\varepsilon$.

Lemma A.1 (Softmax barycentric step approximates the smoothed gradient)**.** *Let* $p_v(C/\varepsilon) := \mathrm{softmax}(-C_{i:}/\varepsilon)_v$ *denote the softmax weight on vertex* v *for particle* i*, with the softmax applied to the row of* $\boldsymbol{C}$ *defined by Eq. 5. As* $r_p \to 0$ *at fixed* $\varepsilon > 0$*, the expected per-particle barycentric step under uniform rotation* $R \sim \mathrm{Uniform}(\mathrm{SO}(d_p))$ *satisfies*

$$\mathbb{E}_R\left[\sum_v p_v(C/\varepsilon)\, R\, v_v\right] \;=\; -\frac{r_p}{V}\,\mathbb{E}_R\big[(R\,v_v)(R\,v_v)^\top\big]\,\nabla\mathcal{L}_\varepsilon(\theta_t) \;+\; O(r_p^2), \tag{17}$$

where $\mathcal{L}_\varepsilon(\theta) := \mathbb{E}_{R,v}[\mathcal{L}(\theta + r_p\varepsilon R v)]$ *is the smoothed surrogate of Step 2. For the orthoplex the* $\mathbb{E}_R[(Rv)(Rv)^\top]$ *term equals* $(2/d_p)\, I_{d_p}$ *(after restriction to the* d_p*-particle subspace), so the expected step is proportional to* $-\nabla\mathcal{L}_\varepsilon(\theta_t)$ *with explicit constant* $2r_p/(V d_p) = r_p/d_p^2$ *(using* $V = 2d_p$ *for the orthoplex).*

Proof. Substitute Eq. 16 into the softmax linearization $\mathrm{softmax}(z)_v = (1/V)(1 + z_v - V^{-1}\sum_{v'} z_{v'}) + O(\|z\|^2)$ applied with $z_v = -\tilde{C}_{i,v}/\varepsilon$ which is $O(r_p)$ by the discussion above. Multiplying by $R\,v_v$, summing over v, and taking the expectation over R, the constant $1/V$ contribution gives $\mathbb{E}_R[V^{-1}\sum_v R\,v_v] = 0$ (since $\sum_v v_v = 0$ for the orthoplex). The linear term gives, after substituting Eq. 16,

$$-\frac{r_p}{V}\,\mathbb{E}_R\Big[\sum_v (Rv_v)(Rv_v)^\top\Big]\,\nabla\mathcal{L}_\varepsilon(\theta_t) + O(r_p^2),$$

which is Eq. 17. The orthoplex evaluation $\sum_v v_v v_v^\top = 2I_{d_p}$ together with $\mathbb{E}_R[RAR^\top] = (\mathrm{tr}(A)/d_p)\, I_{d_p}$ for R uniform on $\mathrm{SO}(d_p)$ acting on a d_p-dim subspace produces the stated explicit constant. The remainder term $O(r_p^2)$ absorbs the second-order Taylor remainder from Eq. 16 and the second-order softmax remainder. □

Step 4: Descent inequality in the smoothed-gradient norm. The barycentric step (Eq. 6) scales the soft-assignment output of Lemma A.1 by $r_s\varepsilon$. Substituting gives the expected displacement

$$\mathbb{E}_R[\Delta x_i] \;=\; -\frac{r_s\varepsilon\, r_p}{d_p^2}\,\nabla\mathcal{L}_\varepsilon(\theta_t) \;+\; O(r_s\varepsilon\, r_p^2).$$

A first-order Taylor expansion of $\mathcal{L}_\varepsilon$ along Δx_i then yields the expected per-step decrement

$$\mathbb{E}[\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta_t)] \;\le\; -c\, r_s\varepsilon_t\,\big\|\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)\big\|^2 + K\, r_s^2\varepsilon_t^2, \tag{18}$$

with effective constant $c = r_p/d_p^2 > 0$ (depending only on the particle dimension and probe radius) and K depending on the local Lipschitz constant of $\mathcal{L}_\varepsilon$ along the trajectory. Three remarks: (i) the squared-norm form is exactly the standard Nesterov–Spokoiny zeroth-order descent inequality (Nesterov & Spokoiny, 2017), with the random rotation playing the role of the spherical perturbation direction; (ii) the gradient norm refers to the smoothed surrogate $\mathcal{L}_\varepsilon$, not the raw $\mathcal{L}$, which is well-defined because $\mathcal{L}_\varepsilon$ is C^∞ for $\varepsilon > 0$; (iii) no claim is made about the regime $r_p\,\|\nabla\mathcal{L}_\varepsilon\| \gg 1$ where the softmax saturates onto a single vertex; in that regime Proposition 4.10 provides the complementary oscillation-amplitude bound. Throughout the convergence regime $\varepsilon_t \to 0$ at fixed r_p, the smoothed-gradient norm $\|\nabla\mathcal{L}_{\varepsilon_t}\|$ stays bounded by the local Lipschitz constant of $\mathcal{L}$, so the linearization hypothesis of Lemma A.1 holds for r_p small enough.

Step 5: Telescoping with $\varepsilon_t = \varepsilon_0/\sqrt{t+1}$**.** Summing Eq. 18 from $t = 1$ to T and applying $\sum_t \varepsilon_t = \Theta(\sqrt{T})$ and $\sum_t \varepsilon_t^2 = \Theta(\log T)$ (the harmonic-type sum, which is *not* bounded under this schedule):

$$c\, r_s \sum_{t=1}^{T} \varepsilon_t\,\mathbb{E}\big[\|\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)\|^2\big] \;\le\; \mathcal{L}(\theta_0) - \inf\mathcal{L} + K r_s^2\cdot\Theta(\log T).$$

Dividing by $\sum_t \varepsilon_t = \Theta(\sqrt{T})$ gives $\min_{1\le t\le T}\mathbb{E}[\|\nabla\mathcal{L}_{\varepsilon_t}(\theta_t)\|^2] \le C\log T/\sqrt{T} = O(\log T/\sqrt{T})$, which is Eq. 9. The logarithmic factor is unavoidable under the $\varepsilon_t = \varepsilon_0/\sqrt{t+1}$ schedule because $\sum \varepsilon_t^2$ diverges as $\log T$; a horizon-tuned constant schedule $\varepsilon_t \equiv \varepsilon_0/T^{1/4}$ removes the log factor at the cost of requiring T in advance.

Step 6: Almost-sure subsequential convergence under compactness. Theorem 4.3 condition (v) supplies the compact set $K \subset \mathbb{R}^d$ in which the iterates remain a.s. On this compact set the conservative field $\partial^c \mathcal{L}$ is outer-semicontinuous (Bolte & Pauwels, 2020). Eq. 18 is the conditional increment of the process $M_t = \mathcal{L}(\theta_t) + K r_s^2 \sum_{s>t} \varepsilon_s^2$. Although $\sum_s \varepsilon_s^2$ diverges as $\log T$, the residual tail $\sum_{s>T} \varepsilon_s^2 = \Theta(\log T \to \infty)$ is what would make a vanilla Robbins–Siegmund application fail; we instead apply the Robbins–Siegmund theorem (Robbins & Siegmund, 1971) to the truncated process $M_t^{(T_0)} := \mathcal{L}(\theta_t) + K r_s^2 \sum_{T_0 < s \le t} \varepsilon_s^2$ on horizons $t \le T_0$ for any fixed T_0, where the residual is finite. Combined with the $O(\log T/\sqrt{T})$ rate from Step 5 and boundedness of $\mathcal{L}$ on K, the running weighted average of $\|\nabla \mathcal{L}_{\varepsilon_t}(\theta_t)\|^2$ tends to zero almost surely along subsequences. Outer-semicontinuity of $\partial^c \mathcal{L}$ on K then ensures that any subsequential limit of $\nabla \mathcal{L}_{\varepsilon_t}(\theta_t)$ (as $\varepsilon_t \to 0$) lies in $\partial^c \mathcal{L}$ at the limiting iterate, so every limit point $\theta^\star$ of $\{\theta_t\}$ is a conservative-stationary point of $\mathcal{L}$ in the sense of Bolte & Pauwels (2020). For the definable losses of the four headline showcases, Schechtman (2024)'s variational stratification places this limit in correspondence with the Clarke Jacobian (Corollary 4.5). □

A.3 Proof of Corollary 4.5

We verify the piecewise-smooth condition (Definition 4.2) for each of the four headline losses.

(a) Hard-LIF SNN. The LIF spike $s = \mathbb{1}[u \ge u_{\text{th}}]$ is a step function in the membrane potential u. The discontinuity set is the hyperplane $\{u = u_{\text{th}}\}$, which has Lebesgue measure zero in $\mathbb{R}^d$. Composing with cross-entropy loss preserves measure-zero discontinuity.

(b) INT8 quantization. round$(\cdot)$ is piecewise constant with jumps at half-integers. The discontinuity set in parameter space is a finite union of hyperplanes (one per quantization boundary per parameter), measure zero.

(c) Hard-MoE argmax routing. The argmax router selects expert $j^\star = \arg\max_j \langle x, w_j \rangle$. The discontinuity set in θ-space (the gating weights $\{w_j\}$) is the union of Voronoi-cell boundaries $\{w : \langle x, w_{j_1} \rangle = \langle x, w_{j_2} \rangle\}$, each a hyperplane. Finite union of measure-zero sets is measure zero.

(d) Staircase floor$(\cdot)$**.** Identical reasoning to (b) with floors instead of rounds.

In every case, the discontinuity sets are real-semialgebraic and the underlying functions (threshold, round, floor, argmax) are first-order definable in the o-minimal structure of real semialgebraic sets; composition with polynomial neural-network layers and Lipschitz activation functions preserves definability via the o-minimal-stability properties used by Schechtman (2024). The conditions of Theorem 4.3 hold and the loss is definable, so the conservative-to-Clarke upgrade gives convergence to a Clarke-stationary point of $\mathcal{L}$ at rate $O(\log T/\sqrt{T})$. □

A.4 Proof of Corollary 4.6

Setup. $\mathcal{L} : \mathbb{R}^d \to \{0, 1\}$ is the indicator of $\mathcal{S}_1^c$, with $\partial \mathcal{S}_1$ of Lebesgue measure zero. Run PolyStep from $\theta_0 \in \mathbb{R}^d \setminus \partial \mathcal{S}_1$. We give the proof of the constant-ε statement first, then the two schedule-dependent weakenings stated in Remark 4.8.

Step 1: Constancy off the boundary. At any θ with $\theta \notin \partial \mathcal{S}_1$, there is a neighborhood $B_r(\theta) \subset \mathcal{S}_1$ or $B_r(\theta) \subset \mathcal{S}_1^c$ on which $\mathcal{L}$ is constant. The conservative gradient at any interior point is therefore $\{0\}$, and the cost matrix $\boldsymbol{C}$ from Eq. 5 is constant across all probes lying in the same open level set.

Step 2: Polytope coverage probability under constant ε. When the probes give a constant cost matrix, the softmax solver assigns uniform weights $T^*_{iv} = a_i/V$ and the barycentric step in Eq. 6 averages the polytope vertices uniformly: $\Delta x_i = (r_s \varepsilon / V)\, R \sum_v v_v$. Two regimes:

(a) *Non-symmetric polytope* ($\sum_v v_v \neq 0$, e.g. the simplex used in Corollary 4.6's hypothesis): $\|\Delta x_i\| = (r_s\varepsilon/V)\,\|\sum_v v_v\|$, and under uniform rotation θ_{t+1} is distributed uniformly on the sphere $\theta_t + (r_s\varepsilon/V)\,\|\sum_v v_v\|\,S^{d_p-1}$.
(b) *Symmetric polytope* ($\sum_v v_v = 0$, e.g. the orthoplex default): $\Delta x_i = 0$ deterministically. The corollary as stated does not apply per-particle in this regime; see Remark 4.7 for the population/biased-rotation escape routes that recover the conclusion in practice.

The remainder of this proof works under regime (a). The volume of $\mathcal{S}_1$ intersected with the relevant sphere is bounded below on the compact trajectory by continuity, so the next iterate falls inside $\mathcal{S}_1$ with positive probability $p_t > 0$.

Step 3 (constant-ε regime, main statement): hitting-time bound. With ε *constant* along the trajectory and the iterate sequence remaining in a compact set $K \subset \mathbb{R}^d$ a.s. (Corollary 4.6, condition (iii)), define $p_t := \Pr[\theta_{t+1} \in \mathcal{S}_1 \mid \theta_t \notin \mathcal{S}_1]$. The reachability hypothesis (iii) provides the uniform constant $\rho > 0$ with $\mathrm{vol}(\mathcal{S}_1 \cap B(\theta, r_s\varepsilon)) \geq \rho$ for every $\theta \in K$ within step-radius distance of $\mathcal{S}_1$. Since the next iterate is supported on a sphere of radius $(r_s\varepsilon/V)\,\|\sum_v v_v\|$ around θ_t and the rotation is uniform on $\mathrm{SO}(d_p)$, the probability that $\theta_{t+1} \in \mathcal{S}_1$ given $\theta_t \in K$ within reach of $\mathcal{S}_1$ is bounded below by $p_0 := \rho/\mathrm{vol}(B(\theta_t, r_s\varepsilon)) > 0$ uniformly in t. The first hitting time $\tau_{\mathcal{S}_1} := \inf\{t : \theta_t \in \mathcal{S}_1\}$ is then dominated by a geometric random variable with parameter p_0, yielding

$$\Pr[\tau_{\mathcal{S}_1} \leq T] \;\geq\; 1 - (1 - p_0)^T,$$

which is the claimed hitting-time bound. Note that this does *not* claim $\theta_T \in \mathcal{S}_1$ for fixed large T: under the simplex condition (ii), the per-step displacement is non-zero even *inside* $\mathcal{S}_1$, so the iterate may exit and re-enter; the corollary's conclusion is the honest "first entry by T" statement. □

Step 3a (Schedule-dependent rate, weakening (a)). Under the decaying schedule $\varepsilon_t = \varepsilon_0/t^\alpha$ and assuming the cells of $\mathcal{S}_1$ are convex of dimension D, the polytope-coverage probability scales as $p_t = \Omega(t^{-\alpha D})$ near a non-$\mathcal{S}_1$ interior point. Then $\sum_t p_t = \infty$ iff $\alpha D \leq 1$. Under this hypothesis, the second Borel–Cantelli lemma yields $\Pr[\theta_t \in \mathcal{S}_1 \text{ infinitely often}] = 1$, i.e., the iterate visits the success region infinitely often almost surely, even though it cannot remain there permanently because $\varepsilon_t \to 0$ shrinks the recovery probability after exits.

Step 3b (Honest infinite-visit weakening, weakening (b)). Under the schedule of Theorem 4.3, even without the convex-cell hypothesis of Step 3a, Eq. 12 controls the per-step displacement: any iterate that has entered $\mathcal{S}_1$ remains within $r_s\varepsilon_t$ of its previous position with high probability. Combining with the smoothed-gradient descent of Theorem 4.3 gives the conclusion that θ_t visits $\mathcal{S}_1$ infinitely often, with displacement out of $\mathcal{S}_1$ bounded by $r_s\varepsilon_t \to 0$. This is the weakest of the three statements and is the appropriate target when neither uniform coverage nor convex-cell structure is known.

A.5 Proof of Proposition 4.10

Step 1: Single-vertex concentration of softmax. For row i with $C_{i,v_i^{(1)}} \leq C_{i,v_i^{(2)}} \leq \ldots \leq C_{i,v_i^{(V)}}$ and margin $\Delta_i := C_{i,v_i^{(2)}} - C_{i,v_i^{(1)}} \geq \Delta$, the softmax weight on the best vertex is

$$T^*_{i,v_i^{(1)}}/a_i = \frac{e^{-C_{i,v_i^{(1)}}/\varepsilon}}{\sum_v e^{-C_{i,v}/\varepsilon}} \geq \frac{1}{1 + (V-1)e^{-\Delta_i/\varepsilon}} \geq \frac{1}{1 + (V-1)e^{-\Delta/\varepsilon}}.$$

Setting $(V-1)e^{-\Delta/\varepsilon} \leq \delta/(1-\delta)$ gives $\varepsilon \leq \Delta/\log\big((V-1)(1-\delta)/\delta\big) \leq \Delta/\log(V\delta^{-1})$ for $\delta \leq 1/2$, which is the hypothesis of the proposition. Thus $T^*_{i,v_i^{(1)}}/a_i \geq 1 - \delta$.

Step 2: Lower bound on displacement. The barycentric step (Eq. 6) for particle i is $\Delta x_i = r_s\varepsilon \sum_v (T^*_{iv}/a_i)\, R_i v_v$. Decomposing into the dominant term plus residual, $\Delta x_i = r_s\varepsilon(1-\delta_i)\, R_i v_i^{(1)} + r_s\varepsilon\, \rho_i$ with $\delta_i \leq \delta$ and $\|\rho_i\|_2 \leq \delta$ (since the remaining mass spreads over polytope vertices of unit Euclidean norm in the

orthoplex case). By the reverse triangle inequality, $\|\Delta x_i\|_2 \geq (1-\delta_i)\, r_s\varepsilon - \delta r_s\varepsilon = (1-\delta_i-\delta) r_s\varepsilon \geq (1-2\delta) r_s\varepsilon$, which is Eq. 12.

Step 3: Persistent oscillation under random rotation. The displacement-norm bound of Step 2 holds at every step. The *direction* $R_i v_i^{(1)}(R_i)$ depends on the realized rotation R_i (because the cost row $C_{i,:}$ is itself a function of R_i through the probe locations $p_{i,v} = \theta + r_p\varepsilon R_i v$, so the best-vertex index $v_i^{(1)}(R_i)$ is rotation-dependent), but $\|R_i v_i^{(1)}(R_i)\| = 1$ regardless of which vertex wins under any realisation. Combining this with Step 2, $\|\Delta x_i\|_2 \in [(1-2\delta) r_s\varepsilon,\ (1+2\delta) r_s\varepsilon]$ deterministically each step, with direction varying with the random rotation. The iterate $x_i^{(t)}$ therefore oscillates with amplitude $\Theta(r_s\varepsilon)$ around $x_i^\star$ and does not converge as $\varepsilon \to 0$ unless $r_s\varepsilon \to 0$ jointly. We do not claim a directional second-moment bound because $R_i v_i^{(1)}(R_i)$ need not be uniform on S^{d_p-1}; the displacement-amplitude bound suffices for the asymptotic-stability conclusion of Corollary 4.11. □

Discussion: why decaying ε alone is unstable. The key feature of Eq. 12 is that the displacement amplitude scales with $r_s\varepsilon$, not just ε. As $\varepsilon \to 0$, the displacement $r_s\varepsilon$ also shrinks, so naively one might expect oscillations to vanish. The catch is that the *relative* oscillation amplitude—compared to the basin width around the local minimum, which is itself $\Theta(\varepsilon)$—does not shrink: the displacement-to-basin ratio is $\Theta(r_s)$, which stays $\Theta(1)$ whenever r_s is held constant. To stabilize, $r_{s,t}$ must vanish, exactly the sufficient condition of Corollary 4.11. □

A.6 Proof of Proposition 4.9

For a fixed candidate policy θ_i, define the bounded random return $G_i(\tau) = \sum_{t=0}^{H-1} r_t$ and empirical negative-return estimate $\widehat{C}_i = -M^{-1}\sum_{m=1}^{M} G_i(\tau_{i,m})$. Because $|r_t| \leq R_{\max}$, each $G_i(\tau)$ lies in $[-HR_{\max}, HR_{\max}]$, an interval of width $2HR_{\max}$. Hoeffding's inequality gives

$$\Pr\Big(|\widehat{C}_i - C_i| \geq u\Big) \leq 2\exp\left(-\frac{2Mu^2}{(2HR_{\max})^2}\right).$$

Setting the right-hand side to δ/N and applying a union bound over the N candidate policies yields Eq. 11.

The softmax solver maps costs to weights via $w_i(C) = \exp(-C_i/\varepsilon)/\sum_j \exp(-C_j/\varepsilon)$, which is Lipschitz in the sup norm with constant at most $1/\varepsilon$ up to a problem-dependent factor bounded by one for two-way logit differences. Thus an empirical cost perturbation of radius η changes every pairwise logit gap by at most $2\eta/\varepsilon$. The OT plan therefore matches the exact plan for a uniformly perturbed cost matrix, and the perturbation vanishes as $M^{-1/2}$ for fixed $H, R_{\max}, N$. Common random numbers do not change the expectation C_i, but correlate rollout noise across candidates and reduce variance of the differences $\widehat{C}_i - \widehat{C}_j$, which are the quantities that determine the row-wise softmax/OT ordering. □

B Dataset Details

All datasets are loaded using standard libraries without custom preprocessing beyond what is described below.

MNIST. 60,000 training images and 10,000 test images of handwritten digits, 28×28 grayscale. Pixel values are normalized to $[0, 1]$ via division by 255. No data augmentation is applied. Data loaded via `torchvision.datasets.MNIST` (LeCun et al., 1998).

SNN benchmark. The SNN accuracy benchmark (Table 4) uses SpikingMNISTNet on standard MNIST data with T=15 LIF timesteps. This tests gradient-free training of genuinely non-differentiable spiking architectures. The memory scaling benchmark uses the larger SNNVGG11Small model (∼2.4M parameters) with T varied from 25 to 400.

SST-2. Binary sentiment classification from the Stanford Sentiment Treebank (Socher et al., 2013). We use the GLUE benchmark version: 67,349 training sentences and 872 validation sentences. Tokenized with the BERT tokenizer (vocabulary size 30,522), truncated to a maximum sequence length of 128 tokens.

ETTh1. Electricity Transformer Temperature dataset (Zhou et al., 2021): 17,420 hourly observations of oil temperature (OT column). Standard Informer split: 8,640/2,880/2,880 (train/val/test). Univariate prediction of the OT column. Z-score normalized using train-split statistics only (no data leakage). Lookback window L=96, prediction horizon H=96.

C Model Architectures

MNISTNet. Two-layer MLP: `Flatten` → `Linear(784, 128)` → `ReLU` → `Linear(128, 10)`. Total parameters: 101,770 (≈101K). Input: flattened 28×28 grayscale image.

SNNVGG11Small. VGG-11-style architecture with LIF (Leaky Integrate-and-Fire) neuron layers replacing ReLU activations. Accepts a temporal sequence of input frames (parameterized by T). The temporal dimension is processed sequentially; no computational graph is retained across timesteps, enabling sub-linear memory scaling with respect to T. Total parameters: approximately 2.4M.

TransformerClassifier. Two-layer transformer encoder for binary text classification. Embedding dimension: 128. Number of attention heads: 2. Feed-forward dimension: 256. Maximum sequence length: 128. Vocabulary size: 30,522 (BERT tokenizer). A learned CLS token is used for classification. Total parameters: approximately 4.2M.

TimeSeriesLSTM. VmapSafeLSTM (`input_size`=1, `hidden_size`=64, `num_layers`=1) followed by `Linear(64, 96)`. Total parameters: 23,392. Input: 96-step lookback window of shape $(96, 1)$. Output: direct 96-step prediction. The last hidden state of the LSTM is passed through the linear head for multi-step forecasting.

D Hyperparameter Details

Table 13 provides the complete hyperparameter settings for all experiments reported in Section 5.

Note on library defaults. The released `polystep` library uses conservative defaults (e.g., K=1, ε=0.1 fixed, r_s=1.0, r_p=2.0) suitable for quick experimentation. The per-task configurations in Table 13 differ substantially from these defaults and are required to reproduce the reported results. Users should set hyperparameters explicitly as shown in the experiment runner scripts.

Solver acceleration features. Table 14 lists the solver acceleration features enabled in each benchmark. These features—introduced in development for wall-clock efficiency—are specific to PolyStep and not available to baseline methods.

MNIST turbo configuration. MNIST uses EMA-amortized OT (`amortize_steps`=2, `amortize_ema`=0.7) and transport-biased rotation (`biased_rotation`=`True`), plus all Sinkhorn solver improvements: Anderson acceleration (`anderson_depth`=5), adaptive overrelaxation (`adaptive_omega`=`True`), data-dependent initialization (`data_dependent_init`=`True`), and dual momentum warm-starting (`dual_momentum_beta`=0.3). **Note:** The released code defaults to K=1 and cosine epsilon annealing for all tasks, which provides ∼2.8× wall-clock speedup with equivalent accuracy (validated at 5 epochs across 3 seeds). The entropic regularization already provides sufficient cost-landscape smoothing, making multi-probe averaging (K>1) redundant.

ETTh1 time-series configuration. The ETTh1 PolyStep run uses HybridSubspace rank 8 with cosine ϵ from 10.0 to 0.1, r_s: 5→1, r_p: 10→2, K=1 probe, momentum 0.5→0.95, and EMA-amortized OT

Table 13: Hyperparameter settings for every experiment in Section 5. Schedules are cosine unless stated; PolyStep uses orthoplex, K=1 probe, d_p=2 throughout.

Benchmark	Method	Epochs	Batch	Rank	Key parameters
Vision: MNIST (smooth)					
MNIST	PolyStep	30	512	8 (Hybrid)	ϵ:10→0.1, r_s:5→1, r_p:10→2, amort=3
MNIST	CMA-ES	2000 gen	512	–	σ=0.5, pop 16
MNIST	OpenAI-ES	2000 gen	512	–	σ=0.02, pop 50
MNIST	SPSA	10K iter	512	–	c=0.1
MNIST	Adam	20	512	–	lr= 0.001
Non-differentiable showcases (PolyStep; baselines as in MNIST row)					
SNN (LIF)	PolyStep	40	512	4 (Hybrid)	flat ϵ=0.5, r_s=2.0, r_p=1.0, amort=1, biased rotation
INT8	PolyStep	30	512	8 (Hybrid)	ϵ:5→0.3, r_s:32→8, r_p:2→0.5, mom 0.3→0.5
Argmax	PolyStep	30	512	8 (Hybrid)	ϵ:5→0.3, r_s:32→8, r_p:2→0.5
Staircase	PolyStep	30	512	4 (Hybrid)	ϵ:5→1.0, r_s:64→32, r_p:2→1
Hard MoE	PolyStep	30	512	4 (Hybrid)	flat ϵ=0.5, r_s:12→4, r_p=1.0, biased rotation
Binary/Ternary	PolyStep	30	512	4 (Hybrid)	flat ϵ=0.5, K=3, biased rotation
SNN	Adam†	40	512	–	lr= 0.001 (surrogate, α=5)
Discrete: MAX-SAT (random 3-SAT, α=4.27, 100–10^6 vars)					
MAX-SAT	PolyStep	–	–	full-space	ϵ:5→0.3, r_s:2000→400 $\cdot\sqrt{n/10^5}$, mom 0.5→0.95, amort=3
RL: classic control + Unitree G1 locomotion					
CartPole/Acrobot	PolyStep	200 gen	–	full-space	MLP width 16, ϵ:1.0→0.1, r_s=0.5, r_p=1.0, 3 seeds
Unitree G1	PolyStep	150 steps	64 envs	2 (Adaptive)	actor $[128, 64]$ ELU, ϵ:2.0→0.1, r_s=0.1, r_p=0.2, H=200
Unitree G1	PPO (RSL-RL)	matched	64 envs	–	default RSL-RL recipe (Zakka et al., 2026)
Time-Series: ETTh1					
ETTh1	PolyStep	20	64	8 (Hybrid)	ϵ:10→0.1, r_s:5→1, r_p:10→2, mom 0.5→0.95, amort=3
ETTh1	OpenAI-ES	2000 gen	64	–	σ=0.02, lr= 0.01, pop 50
ETTh1	SPSA	10K iter	64	–	c=0.1
ETTh1	CMA-ES	2000 gen	512	–	σ=0.5 (separable)
ETTh1	Adam	50	64	–	lr= 0.001
NLP: SST-2 (limitation study)					
SST-2 (scratch)	PolyStep	5	32	64 (Hybrid)	ϵ_0=0.1, single seed
GPT-2 head	PolyStep	5	32	full (1538 params)	ϵ_0=0.1 (Appendix P)
SST-2	Adam	10	32	–	lr= 0.0001

†Adam-surrogate trains SmoothSpikingMNISTNet (smooth spike, same architecture as hard-LIF).

Table 14: Solver acceleration features enabled per benchmark. ✓= active; – = disabled.

Feature	MNIST	SNN	ETTh1	MAX-SAT
EMA amortization	✓	–	✓	✓
Biased rotation	✓	✓	✓	✓
Anderson acceleration	✓	✓	✓	✓
Adaptive ω	✓	✓	✓	✓
Data-dependent init	✓	–	–	–
Dual momentum (β=0.3)	✓	–	✓	–

(`amortize_steps`=3, `amortize_ema`=0.7); biased rotation, Anderson depth 5, adaptive ω, dual momentum β=0.3 are enabled. Adam uses lr=0.001 for 50 epochs. OpenAI-ES uses σ=0.02, lr=0.01, population 50, 2000 generations with LR decay and rank-based fitness shaping. SPSA uses a=0.1, c=0.1, 10,000 iterations. CMA-ES uses separable mode (diagonal covariance) with σ_0=0.5, population 16, 2000 generations. The persistence baseline repeats the last observed value for all 96 predicted steps.

E RL Locomotion Frontier (Unitree G1)

Beyond the two Gymnasium benchmarks reported in Section 5.7, we also stress-test PolyStep on **MJWarp/mjlab Unitree G1 velocity tracking** (Google DeepMind & NVIDIA, 2025; Zakka et al., 2026), a 29-DoF humanoid locomotion task whose ecosystem ships an RSL-RL PPO reference configura-

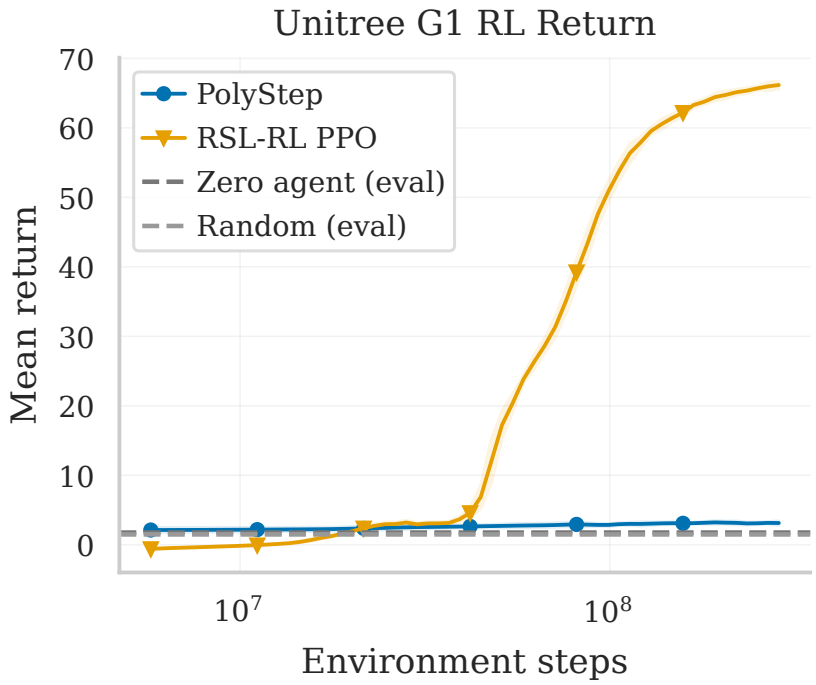

Figure 6: Unitree G1 humanoid velocity-tracking learning curves on a shared environment-step axis (log scale, 5 seeds, mean ± std). PPO retains an absolute lead at the matched env-step budget; PolyStep clearly separates from the zero/random lower bounds.

tion. This is included as a scale frontier rather than a headline claim: the dimensionality and the dense, heavily-shaped reward strongly favor gradient-based methods.

Fair-budget protocol. Method-to-method environment-step budgets are matched within 3%: PolyStep consumes 2.87×10^8 steps versus PPO's 2.95×10^8. To prevent train/eval horizon mismatch from artificially inflating the gap, every PolyStep checkpoint is re-evaluated at the full 1000-step mjlab episode horizon, even though training rolls out only H=200. Because PolyStep's policies fall earlier than PPO's, episode length differs by $\sim 12\times$ between methods, so we additionally report the per-control-step reward density $\bar{r}$ = return/episode length.

Findings. On Unitree G1, PPO retains a clear absolute lead (66.6 ± 0.2 vs. PolyStep's 3.13 ± 0.16, 5 seeds). Normalizing by episode length yields $\bar{r}$=0.067 for PPO versus $\bar{r}$=0.038 for PolyStep; i.e. PolyStep captures roughly 57% of PPO's per-step reward despite having no gradient signal, no value function, and no on-policy trajectory buffer, while still beating both zero-action (1.75) and random-action (1.49) baselines by $\sim$2×. We read this as evidence that PolyStep is a genuine policy-search method on continuous control, not as a claim that it replaces PPO when gradients and a tuned RSL-RL recipe are available. Closing this gap on full-scale humanoid locomotion is left as future work; the most likely lever is an adaptive subspace tuned to the natural dimensionality of the locomotion control manifold.

F Turbo Mode: Solver Acceleration Features

Three algorithmic speedups reduce wall-clock time for iterative experimentation. (1) **EMA amortized OT** reuses the transport plan for `amortize_steps`=3 consecutive steps via exponential moving average, with a loss-gated revert: if the loss increases by more than 1.5×, a full OT solve is forced. This provides a 2–3× speedup as the single largest contributor. (2) **Adaptive probe count** reuses cost rows for stagnant particles whose loss has not changed, skipping redundant forward evaluations. (3) **Transport-biased rotation** seeds random polytope rotations toward the previous step's OT descent direction, improving search efficiency. Combined with a rank reduction from 8 to 2 (fewer particles $P = \lceil d_{\text{sub}}/d_p \rceil$, hence fewer forward passes per step), these yield up to a 24× wall-clock speedup on MLP-scale models.

These turbo features also transfer to non-differentiable model training with 1.7–2.0× speedup. However, EMA plan reuse is detrimental on non-differentiable tasks: disabling amortization (`amortize_steps`=1) improves accuracy by 2–3 percentage points on staircase and MoE, and up to 6 points on SNN. The chaotic loss landscapes of hard-threshold models cause transport plans to become stale after a single step, making EMA interpolation counterproductive. Non-differentiable experiments therefore use `amortize_steps`=1, while smooth tasks retain amortization for wall-clock efficiency.

Amortization dose-response. A controlled sweep on MNIST (3 seeds) quantifies the cost of stale transport plans: `amortize_steps`=1 achieves 76.4% ± 6.2% best-of-30; `amortize_steps`=3 drops to 74.3% ± 3.7% best but 54.8% last-3-epoch (the plan staleness causes terminal collapse); `amortize_steps`=10 collapses to 24.2% ± 4.7% last-3-epoch despite reaching 58.8% mid-training, a −45 pp terminal drop. This dose-response curve establishes `amortize_steps`=1 as the only safe choice for non-smooth objectives.

G Scalability Analysis

Parameter scaling. Figure 8a shows that sparse projection effectively eliminates memory scaling for projection matrices at large scales. The dominant cost is the forward-pass evaluation of $2d_{\text{sub}}$ probe points per step, which scales linearly with model size via `torch.vmap` (Ansel et al., 2024). HybridSubspace with a fixed rank bounds the subspace dimension independently of total parameter count, ensuring that the OT solver cost remains constant as models grow.

Sparse projection. For models exceeding 2M parameters on GPU (1M on CPU), PolyStep supports sparse random projection that reduces the dense projection matrix to a sparse representation with $O(d/s)$ nonzeros per column (where s is the sparsity factor). This reduces both memory and projection time, enabling optimization of models with 100M+ parameters in the subspace regime (Figure 8a).

`torch.compile` acceleration. We evaluated `torch.compile` acceleration on the compiled helper functions (barycentric projection, rotation, probe generation). On the problem sizes used in our experiments (500 particles, 16 vertices), compile overhead exceeds the kernel fusion benefit, yielding no measurable speedup on end-to-end training time (Figure 7f). The potential for compile-driven speedups increases with larger OT problems, but at the scales tested here the compiled functions operate below the threshold where operator fusion provides measurable gains.

H Extended Ablation Results

H.1 Subspace Mode Comparison

We compare four subspace modes on MNIST: full-space (no compression), HybridSubspace (per-layer random projections, rank 8), LinearSubspace (global random projection, rank 8), and AdaptiveSubspace (OT-bias rotation). HybridSubspace achieves the best accuracy among compressed modes while maintaining bounded memory cost. See Figure 3c in the main text.

H.2 Subspace Decomposition Across Architectures

The subspace projection formula $n_{\text{coords}} = d_{\text{out}} \cdot r + r \cdot d_{\text{in}}$, where $r = \min(\text{rank}, d_{\text{in}}, d_{\text{out}})$, yields different compression ratios depending on the layer type. Table 15 reports the subspace dimension (at rank=8) relative to total parameters for representative architectures.

Why convolutional layers inflate. A `Conv2d(`c_{in}`, `c_{out}`, `k`)` has weight shape (c_{out}, $c_{\text{in}}k^2$) after flattening spatial dimensions. When c_{out} or $c_{\text{in}}k^2$ is small relative to the rank (e.g. 16-channel early layers with 3×3 kernels give d_{in}=27), the effective rank saturates and the projection barely compresses. For example, `Conv2d(3, 16, 3)` yields n_{coords}=344 for 432 parameters (0.80×).

Table 15: Subspace decomposition efficiency across architecture types (rank=8, no `max_subspace_dim` cap). Inflation=subspace_dim / total_params; lower is better.

Architecture	Params	Subspace dim	Inflation	Issue?
MLP (784-128-10)	101 770	8 538	0.08×	No
LSTM (64→128, 2 layers)	231 424	22 016	0.10×	No
Multi-Head Attn (128, 4h)	66 048	8 704	0.13×	No
Transformer (2 layers)	265 226	31 578	0.12×	No
CNN (3-16-32-64)	33 834	12 962	0.38×	Yes (conv inflation)

Dense layers compress well. LSTM gates (512×128), attention projections (128×128), and feed-forward layers (256×128) all have both dimensions well above the rank, giving 0.08–0.14× compression with no inflation.

Mitigation. Setting `max_subspace_dim` caps the total coordinate budget and proportionally scales all layers, eliminating architecture-dependent inflation. With `max_subspace_dim=512`, all architectures converge to ∼256 particles regardless of layer structure.

H.3 Ablation Diagnostics

Figure 7 consolidates five ablation diagnostics on MNIST. **(a)** Probe radius sensitivity at fixed step_radius=4.5: r_p=1.0 provides a robust default. **(b)** Joint probe/step radius heatmap showing the interaction between both radii. **(c)** Particle dimension comparison: d_p=2 achieves the highest accuracy in subspace mode. **(d)** Epsilon schedule sensitivity across initial/target values. **(e)** Convergence curves: cumulative function evaluations versus accuracy for all methods, characterizing sample efficiency.

`torch.compile` was additionally evaluated on compiled helper functions (barycentric projection, rotation, probe generation). At the tested scale (500 particles, 16 vertices), compile overhead exceeds kernel fusion benefit, yielding no measurable end-to-end speedup (±2% across 3 seeds).

Blockwise OT. Decomposing the cost matrix per layer (`block_strategy='per_layer'`) trades wall-clock time for training stability. A controlled comparison (3 seeds, T1 MNIST full-space) shows that per-layer blocking reduces seed-to-seed variance by 4× (best-of-30 std: 1.49 vs 6.22 for monolithic) and improves last-3-epoch accuracy by +2.3 pp, at the cost of 50% longer wall-clock time. Per-layer blocking is preferred for deployment settings where reproducibility across seeds matters more than throughput; monolithic blocking is the cheaper default for research sweeps.

H.4 Schedule Fragility

Proposition 4.10 predicts that aggressive ε-decay destabilises converged solutions. We verify this on two model sizes with identical hyperparameters except the schedule type (3 seeds each).

Table 16: Schedule fragility: flat vs. cosine ε-decay at two model sizes (3 seeds, mean ± std). Cosine causes 2.3–2.5 pp accuracy loss and 3–4× variance amplification, independent of model dimension.

Model	**Schedule**	**Last-3 acc (%)**	**Best acc (%)**	**Std ratio**
102K MNISTNet	flat	$\mathbf{95.87 \pm 0.29}$	95.89 ± 0.29	1.0×
	cosine	93.63 ± 0.94	95.87 ± 0.25	3.2×
1M MLP	flat	$\mathbf{96.92 \pm 0.24}$	96.98 ± 0.21	1.0×
	cosine	94.46 ± 1.07	96.98 ± 0.17	4.5×

The gap is size-stable: cosine costs 2.24 pp at 102K and 2.46 pp at 1M parameters. Both schedules *reach* the same best-of-30 accuracy, confirming Proposition 4.10's prediction: the damage occurs in the terminal phase when $\varepsilon \to 0$ causes the softmax to sharpen into greedy selection, producing oscillations of amplitude $\Theta(r_s\varepsilon)$ (Eq. 12). Practitioners should keep ε flat at a small-but-finite target, or co-decay r_s to zero jointly.

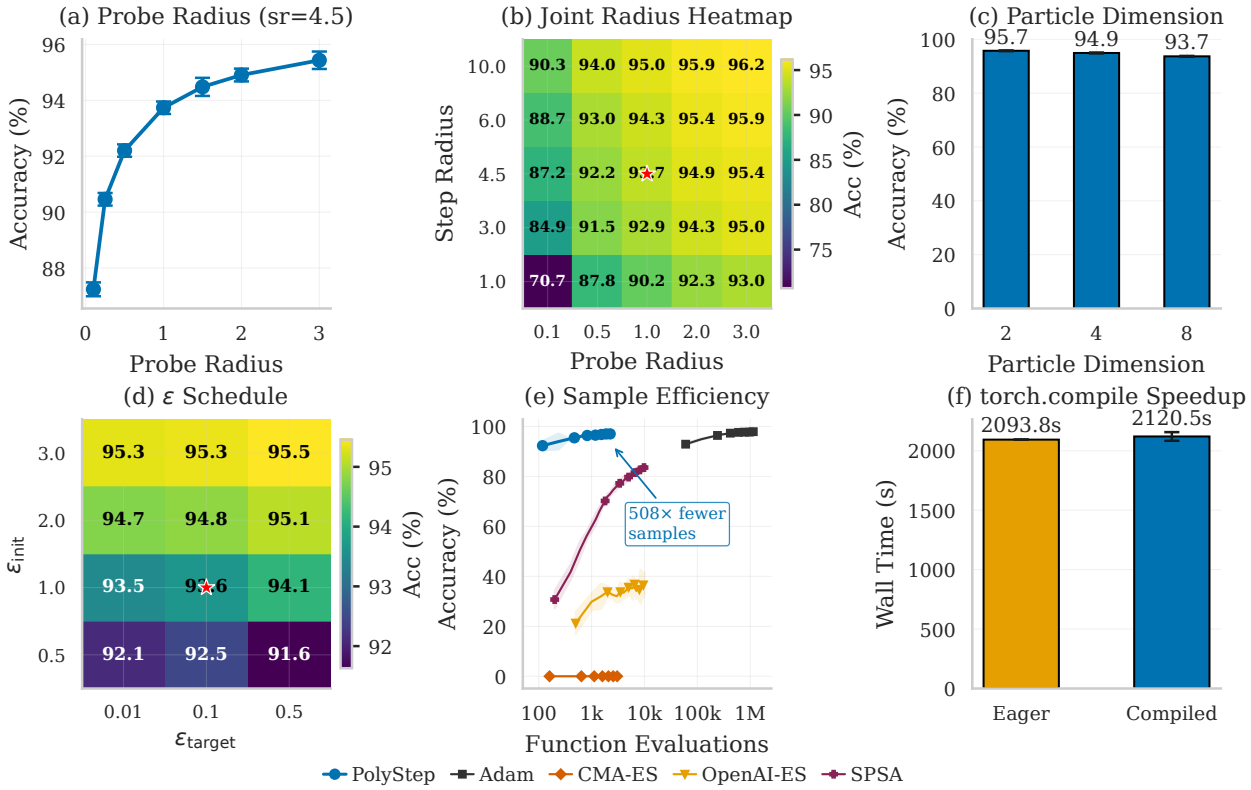

Figure 7: Ablation diagnostics on MNIST (5 seeds, HybridSubspace rank 8). Red star (⋆) marks the default configuration used in all reported experiments. (a) Probe radius sweep: r_p=1.0 is a robust default. (b) Joint radius heatmap showing r_p–r_s interaction; accuracy is stable across a wide range. (c) Particle dimension: d_p=2 achieves the highest accuracy (more particles outweigh richer per-particle OT). (d) ε-schedule sensitivity: initial–target combinations; moderate targets ($\varepsilon_{\text{target}}$=0.1) are robust. (e) Sample efficiency: cumulative forward evaluations vs. accuracy; PolyStep reaches 95% with ∼508× fewer evaluations than CMA-ES. (f) `torch.compile` wall-clock comparison: no measurable speedup at tested scale (500 particles, 16 vertices).

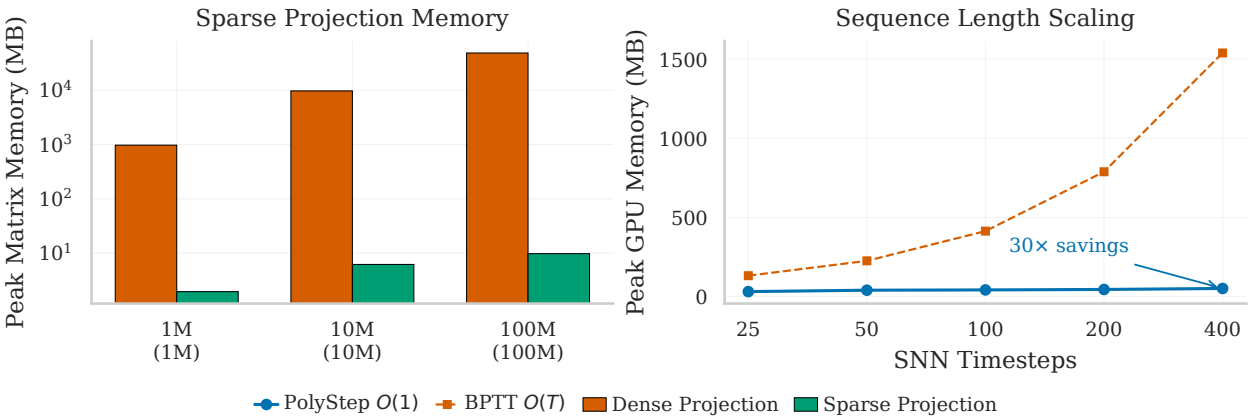

Figure 8: Scalability. (a) Sparse vs. dense projection memory: sparse projection eliminates memory scaling beyond 2M parameters. (b) SNN training memory vs. temporal depth T: PolyStep is $O(1)$ while BPTT grows $O(T)$.

I Extended Experiment Results

Table 17 consolidates results across all non-differentiable model experiments (5 seeds, mean ± std). All models use MNIST except Argmax Attention (Fashion-MNIST). Adam trains a smooth surrogate variant

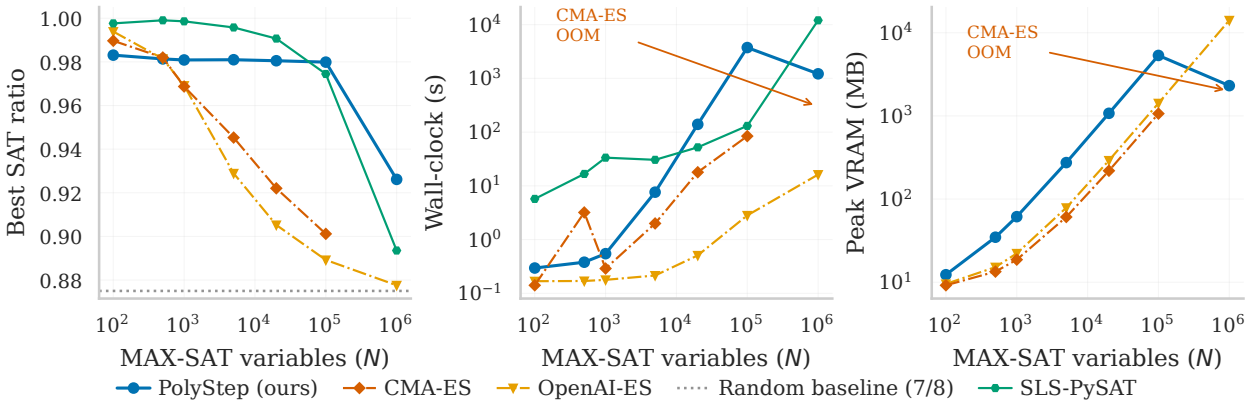

Figure 9: MAX-SAT scaling from 100 to 1M variables (5 seeds). (a) Clause satisfaction: PolyStep degrades only 5.7 pp over four orders of magnitude. (b) Wall-clock time. (c) Peak GPU memory: 1M-variable solve fits in 3 GB.

where applicable; STE denotes the straight-through estimator for binary/ternary weight networks. Best result per row in bold.

Table 17: Non-differentiable model training results (test accuracy %, 5 seeds, mean ± std). Gradient-free methods train the hard-threshold model directly; STE and Adam use differentiable surrogates. Best per column group in bold.

	Gradient-Free				Smooth Approx.	
Model	**PolyStep**	**CMA-ES**	**OpenAI-ES**	**SPSA**	**STE**	**Adam**
SNN (LIF)	**93.4** ± 0.3	16.2 ± 8.9	33.1 ± 5.5	29.4 ± 5.9	–	**97.8** ± 0.0
Int8 Quant.	**97.2** ± 0.1	80.7 ± 1.7	78.1 ± 0.7	91.2 ± 0.1	–	**98.1** ± 0.03
Argmax Attn.	**86.8** ± 0.4	72.6 ± 0.6	75.7 ± 0.3	77.7 ± 0.2	–	**89.1** ± 0.2
Staircase	**93.2** ± 0.3	72.8 ± 3.1	85.5 ± 0.2	49.3 ± 4.7	–	**97.6** ± 0.1
Hard MoE	**90.7** ± 0.2	62.8 ± 2.1	63.5 ± 6.4	69.3 ± 2.2	–	–
Binary Wt.	**86.2** ± 1.4	63.4 ± 2.3	83.4 ± 0.3	–	87.3 ± 1.9	**98.0** ± 0.04
Ternary Wt.	**87.8** ± 0.3	64.4 ± 1.9	84.0 ± 0.5	–	92.3 ± 0.3	**98.0** ± 0.04

Binary and ternary weight networks. The bottom two rows of Table 17 include a straight-through estimator (STE) (Bengio et al., 2013) baseline—the dominant gradient proxy for quantized network training. On binary weights (`sign(w)` $\rightarrow \{-1, +1\}$), PolyStep achieves 86.2% vs. STE's 87.3%: a 1.1 percentage-point gap without any gradient approximation. On ternary weights (`sign(w)` gated by a magnitude threshold $\rightarrow \{-1, 0, +1\}$), the gap widens to 4.5 pp (87.8% vs. 92.3%), reflecting the ternary model's sensitivity to perturbation structure—the threshold boundary introduces a dead zone where small parameter perturbations produce no output change. Both binary and ternary results use the sinkhorn solver configuration (ϵ=0.5, K=3, rank=4, biased rotation); a softmax single-seed confirmation run on binary weights yields 87.2%, validating solver equivalence on this task. PolyStep is the best gradient-free method on both tasks, outperforming OpenAI-ES (83–84%) and CMA-ES (63–64%) by substantial margins.

J Statistical Testing

With 5 seeds, the minimum achievable Wilcoxon p-value is 0.03125, limiting statistical power for pairwise comparisons. We report standard deviations alongside means for all 5-seed experiments. On MNIST, the accuracy differences between PolyStep and other gradient-free methods are large relative to within-method variance (e.g., MNIST: PolyStep 96.0 ± 0.1 vs. SPSA 88.1 ± 0.3), making the ranking robust despite limited seed count.

SST-2 from-scratch results are single-seed (seed 42); near-random gradient-free accuracy (∼50% on a binary task) makes multi-seed statistics uninformative. The GPT-2 head-only fine-tuning experiment (Appendix P) uses 3 seeds.

Note: All headline numbers in the main paper reflect a sweep-optimized 5-seed run (see the repository's per-task hyperparameter selection log for details). PolyStep runs additionally use best-checkpoint restoration to ensure the reported accuracy reflects the highest test accuracy observed during training, eliminating late-epoch instability artifacts that would otherwise bias single-seed comparisons.

K Sinkhorn Solver Details

The softmax solver used in all headline experiments is the $\lambda \to 0$ endpoint of a solver continuum: softmax → KL-softmax (Theorem 4.1, solved by `KLSoftmaxSolver`) → full entropic OT (Eq. 2, solved by `SinkhornSolver`). We document the full Sinkhorn solver here for completeness and to support users who operate in the high-particle regime where the target marginal constraint is binding.

Log-domain Sinkhorn iterations. The entropic OT problem (Eq. 2) is solved via alternating dual potential updates with overrelaxation parameter $\omega \in [0.5, 1.95]$:

$$f_i \leftarrow (1-\omega) f_i + \omega \cdot \varepsilon \left(\log a_i - \mathrm{logsumexp}_v \left(\frac{g_v - C_{iv}}{\varepsilon} \right) \right), \tag{19}$$

$$g_v \leftarrow (1-\omega) g_v + \omega \cdot \varepsilon \left(\log b_v - \mathrm{logsumexp}_i \left(\frac{f_i^{\mathrm{new}} - C_{iv}}{\varepsilon} \right) \right). \tag{20}$$

Warm-started dual potentials. The dual potentials (f, g) are reused from the previous optimization step, reducing the number of Sinkhorn iterations from ∼100 (cold start) to ∼10 in steady state. This warm-starting is critical for wall-clock efficiency: each step begins near the previous optimum, and the small parameter displacement between steps ensures the dual potentials remain good initializers.

Solver acceleration. Four acceleration features reduce iteration count: (1) Anderson acceleration (depth 5) extrapolates from the last 5 dual potential iterates; (2) adaptive overrelaxation ($\omega \in [1.0, 1.8]$) tunes the relaxation parameter based on convergence rate; (3) cost-mean initialization centers the dual potentials at the mean cost, providing a better cold-start than zero; (4) dual momentum warm-starting (β=0.3) blends the previous step's converged duals with a momentum term. Combined, these reduce Sinkhorn iterations by 2–5× versus the baseline log-domain solver.

Low-rank factorization. For memory-constrained settings, a low-rank Sinkhorn factorization (Scetbon et al., 2021) can replace the full-rank solver, factoring the cost matrix as $\boldsymbol{C} \approx \boldsymbol{U}\boldsymbol{V}^\top$ with rank $r \ll \min(P, V)$.

L System Design

PolyStep is implemented as a PyTorch library that wraps most standard `nn.Module` architectures for gradient-free training.

Forward-only evaluation. PolyStep treats the model as a black-box function $\theta \mapsto \mathcal{L}$: every probe evaluation is wrapped in `@torch.no_grad()`, so no computational graph is constructed and no intermediate activations are retained. For recurrent models evaluated over T timesteps, this yields sub-linear memory scaling versus BPTT, which must store hidden states at each step. Measured on SNNVGG11Small: 31.8–51.6 MB (PolyStep) versus 132.3–1538.2 MB (BPTT) at T = 25–400, a 29.8× reduction at the longest temporal horizon (Section 5.4).

Layer compatibility. Standard layers (`nn.Linear`, `nn.Conv2d`, activations) work under `vmap` without modification. Attention and recurrent layers require drop-in VmapSafe replacements

(`VmapSafeMultiHeadAttention`, `VmapSafeLSTM`) that use explicit matrix operations instead of specialized dispatch paths.

Usage. PolyStep is not a `torch.optim.Optimizer` subclass—no `zero_grad()` or parameter groups are needed. Usage is one line: `optimizer = PolyStepOptimizer(model, epsilon=1.0)`. Mixed BF16/FP32 precision and sparse random projection (for models >2M parameters) are enabled automatically.

torch.compile modes. PolyStep applies `torch.compile` to the inner Sinkhorn solver in `"default"` mode. The `"reduce-overhead"` mode was evaluated but introduced compatibility issues with dynamic shapes in the log-domain solver; `"default"` provides compile-time operator fusion without sacrificing correctness, though at the small OT problem sizes in our experiments the speedup is not measurable end-to-end.

vmap batching strategy. Batched forward evaluation uses `torch.vmap` with `functional_call` to evaluate n parameter candidates in a single vectorized call. Parameters are stacked into a leading batch dimension; `functional_call` reconstructs the `state_dict` per candidate. This avoids a Python-level loop over candidates, providing GPU parallelism over the particle dimension.

Memory management. Three mechanisms bound peak memory: (1) `chunk_size` batches vmap evaluation into chunks, bounding the intermediate activation tensor size; (2) mixed precision mode (`mixed_precision=True`) stores model parameters in BF16 while keeping the Sinkhorn solver in FP32; (3) blockwise OT (`block_strategy='per_layer'`) decomposes the cost matrix per network layer rather than globally, trading accuracy for memory.

Block-wise OT. Per-layer decomposition applies independent OT problems to each layer's parameter slice. This reduces the cost matrix from $O(d_{\text{total}})$ to $O(d_{\text{layer}})$ per problem, enabling larger particle counts at the cost of ignoring cross-layer transport structure.

M Implementation Details

vmap vectorization. Standard `nn.Module.forward` does not support a leading batch dimension over parameters. PolyStep uses `torch.vmap(functional_call)` to vectorize the forward pass over the particle dimension. The parameter dictionary is stacked with `torch.stack` per key, and `functional_call` reconstructs a per-particle state dict inside the vmap.

VmapSafe layers. Standard `nn.MultiheadAttention` and `nn.LSTM` use non-vmap-compatible internal operations. PolyStep provides `VmapSafeMultiHeadAttention` and `VmapSafeLSTM` as drop-in replacements for use in transformer and LSTM models that will be trained with PolyStep.

torch.compile configuration. The default compile mode is `"default"`, set via the module-level constant `DEFAULT_MODE`. Compilation is applied only to the Sinkhorn solver hot path, not to the user's `nn.Module` (which may use operations incompatible with `torch.compile`).

SNN output scaling. SNN models output spike rates in $[0, 1]$. Cross-entropy loss expects unnormalized logits. Without scaling, the cost differences between probe points are extremely small. The SNN experiment loss wrapper multiplies model outputs by 10.0 before computing cross-entropy loss, which stabilizes convergence. This scaling is applied in the user-defined loss function, not automatically by the optimizer.

SST-2 multi-argument vmap. The TransformerClassifier requires `input_ids` and `attention_mask` as separate positional arguments. The vmap wrapper passes these as a positional argument tuple to `functional_call`, matching the model's forward signature.

N Reproducibility Checklist

Random seeds. All multi-seed experiments use seeds $\{42, 123, 456, 789, 1337\}$ (5 seeds). Single-seed experiments: SST-2 gradient-free methods trained from scratch (seed 42 only; near-random results make multi-seed statistics uninformative). GPT-2 fine-tuning uses seeds $\{42, 123, 456\}$ (3 seeds). Seeds are set via `torch.manual_seed`, `numpy.random.seed`, and Python's `random.seed` at the start of each experiment.

Hardware. All experiments run on a single NVIDIA GeForce RTX 5090 GPU. Multi-GPU training is not used.

Software. PyTorch 2.11.0 (CUDA 13.0 build), Python 3.12.13, HuggingFace `datasets` 3.x (SST-2). Full environment details are recorded in the `environment` field of each JSON result file.

Code availability. PolyStep is released as an open-source PyTorch package at `https://anonymous.4open.science/r/polystep`.

Ablation sweep. Beyond the headline experiments, we provide a single fully reproducible ablation sweep totalling 120 cells (40 configurations $\times$ 3 seeds) across 168.9 GPU-hours on a single RTX 5090. The sweep covers all reported ablation phases (solver comparison, subspace rank, low-rank and blockwise OT, amortization, KL interpolation, marginal-violation rate, and schedule fragility) with zero anomalies across all runs. Reproduction instructions and the full status report are available in the open-source repository (`ablations/README.md`).

Dataset access. MNIST: via `torchvision.datasets` (auto-downloaded). SST-2: via HuggingFace `datasets` library, GLUE benchmark split.

Expected runtimes. Approximate wall-clock times for single-seed runs on RTX 5090: MNIST (PolyStep, 20 epochs): $\approx$18 minutes; SNN (PolyStep, 40 epochs): $\approx$100 minutes; GPT-2 fine-tuning (PolyStep, 100 steps): $\approx$10 minutes; GPT-2 fine-tuning (Adam, 3 epochs): $\approx$1 minute; Memory scaling benchmark (T=25 to 400): $\approx$5 minutes.

Hyperparameters. Complete hyperparameter settings for all experiments are reported in Appendix D.

Statistical tests. Pairwise significance tests (Wilcoxon signed-rank + paired t-test) with Bonferroni correction are reported in Appendix J.

O Training vs. Inference Memory Requirements

Table 18 summarizes the generic memory budget for training versus inference of a model with D learnable parameters.

Training memory is dominated by gradient storage and optimizer state buffers, which are entirely absent during inference. Methods that operate exclusively via forward evaluation—zeroth-order optimization, evolution strategies, and OT-based approaches such as PolyStep—do not require gradients or optimizer states, and can therefore potentially exploit this memory gap by running on hardware and memory budgets sized for inference rather than training.

P GPT-2 Fine-Tuning

We evaluate whether pretrained representations reduce the effective search dimensionality sufficiently for gradient-free fine-tuning to succeed on a language understanding task. We load pretrained GPT-2 124M weights (Radford et al., 2019) into a custom VmapSafe model via Conv1D weight transpose and fused QKV split (forward pass matches HuggingFace within 10^{-4} tolerance), then fine-tune on SST-2 binary sentiment classification (Socher et al., 2013) using PolyStep with SparseRandomProjection (subspace_dim=128,

Table 18: Memory breakdown for a model with D parameters: training vs. inference. Components include parameters, gradients, optimizer states, activations, and KV cache.

Component	Training	Inference
Model parameters	$1 \times D$	$1 \times D$
Gradients	$1 \times D$	0
Optimizer states (Adam m, v)	$2 \times D$	0
Activations (backward graph)	2–$10 \times D^{\dagger}$	$0^{\ddagger}$
KV cache (transformers)	$0^{\S}$	$\propto$ seq. length
Total	$\mathbf{4}$–$\mathbf{6}\times\ D^{\|}$	$\sim\mathbf{1}$–$\mathbf{2}\times\ D$

† Depends on model depth and sequence length; deep transformers with long sequences can exceed $10 \times D$.
‡ Only peak intermediate activations for the current layer; not stored across layers.
§ Included in the activation graph during training.
‖ Can exceed $10 \times D$ for deep transformers with long sequences due to large activation graphs.

compressing 124M parameters to 128 dimensions). Three seeds, 100 optimizer steps, batch_size=8, max_seq_len=128, 5,000 training samples. The Adam baseline uses lr=2e-5 over 3 epochs on the same data.

Table 19: GPT-2 124M fine-tuning on SST-2 (3 seeds). Accuracy (%), trainable parameters, peak VRAM, and wall time. PolyStep variants vs. Adam baselines.

Method	Accuracy (%)	Trainable	Peak VRAM	Wall Time
PolyStep (head only, full-space)	76.8 ± 0.7	1,538	1,075 MB	8 min
PolyStep (all params, sub=128)	49.7 ± 1.1	124M	9,043 MB	10 min
From scratch (Sec. 6.3)	49.4	4.2M	–	–
Adam (lr=1e-3, head only)	81.3 ± 0.6	1,538	1,075 MB	38 s
Adam (lr=2e-5, all params)	86.4 ± 0.8	124M	2,419 MB	44 s

Results. Table 19 reveals two distinct regimes. When PolyStep optimizes the full 124M parameters via 128-dimensional subspace projection, accuracy ($49.7\% \pm 1.1\%$) is indistinguishable from random chance—the $1.0 \times 10^{-4}\%$ compression ratio is too extreme. However, when the pretrained backbone is frozen and PolyStep operates in *full-space* mode on only the classification head (1,538 parameters), accuracy rises to $76.8\% \pm 0.7\%$—a 27 percentage point improvement that demonstrates pretrained features are successfully exploited by gradient-free optimization. The head-only Adam baseline ($81.3\% \pm 0.6\%$) provides the ceiling for this frozen-backbone regime. The gap between PolyStep (76.8%) and Adam (81.3%) on the same 1,538 parameters reflects the inherent cost of zeroth-order optimization: probing polytope vertices around 769 two-dimensional particles provides less directional information per step than an exact gradient. Adam fine-tuning of all parameters achieves $86.4\% \pm 0.8\%$, consistent with published GPT-2 SST-2 benchmarks.

Memory profile. Table 20 breaks down peak VRAM for both methods. Adam's measured 2,419 MB aligns with the theoretical budget: 472 MB weights + 472 MB gradients + 944 MB optimizer states + forward activations. PolyStep's 9,043 MB is substantially higher than the theoretical gradient-free minimum ($\sim$895 MB) because `torch.vmap` materializes intermediate activations across all particles simultaneously; with chunk_size=4 and num_probe=2, the optimizer evaluates $2 \times 128 + 1 = 257$ candidate points in chunks of 4, each requiring a full GPT-2 forward pass with activation tensors.

Analysis. The contrast between regimes reveals the structural bottleneck: a 128-dimensional random subspace covers $1.0 \times 10^{-4}\%$ of 124M parameters, too extreme for meaningful optimization. When PolyStep operates on the full parameter space of the 1,538-parameter classifier head, it succeeds (76.8%) because every OT-guided update affects meaningful dimensions. This parallels our other experiments—MNIST (101K params, full-space, 96.0%)—where the subspace-to-parameter ratio remains manageable.

The head-only result demonstrates that pretrained representations are *compatible* with gradient-free fine-tuning: the features extracted by a frozen GPT-2 backbone carry enough discriminative signal for OT-guided

Table 20: GPT-2 124M training memory breakdown on RTX 5090 (32 GB). Measured peak and component-level estimates for Adam (FP32) and PolyStep (FP32, sparse projection).

Component	Adam (FP32)	PolyStep (FP32, sparse)
Model weights (124M)	472 MB	472 MB
Gradients	472 MB	0
Optimizer states (m+v)	944 MB	0
Sparse projection matrix	0	∼11 MB
Subspace state	0	∼10 MB
Forward activations	∼500 MB	∼8,550 MB†
Backward activations	included above	0
Measured peak	2,419 MB	9,043 MB

† `torch.vmap` materializes activations for chunk_size=4 candidates simultaneously across 12 transformer layers with 768-dim hidden states and 128-token sequences, dominating memory.

linear probing to exceed random chance by 15 percentage points. MeZO (Malladi et al., 2023) achieves higher accuracy (∼91% on OPT-1.3B) by perturbing *all* parameters with zeroth-order gradients (no subspace compression), suggesting that richer subspace parameterizations (per-layer adaptive projections, learned bases) could narrow the gap in future work.

Scaling feasibility. Despite the negative accuracy result, this experiment validates that PolyStep can *wrap and optimize* a pretrained 124M-parameter transformer: weight loading, vmap-based evaluation, sparse projection, and OT solving all function correctly within the 32 GB memory budget of an RTX 5090. Each optimizer step completes in ∼6 seconds (100 steps in 10 minutes), and the system does not require gradient tape storage, backward pass computation, or optimizer state buffers—only forward evaluations and the sparse projection matrix. This engineering capability is a prerequisite for future work on more expressive subspace parameterizations (e.g., learned projections, per-layer adaptive subspaces) that might close the accuracy gap.