CACHED BOUNDARY TRANSPORT FOR EFFICIENT ACTIVITY INFERENCE IN DEEP PREDICTIVE CODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Predictive coding (PC) networks infer their hidden activities before every weight update, and in deep networks this inner optimization becomes the bottleneck of training. We show that the difficulty is geometric rather than iterative: the activity landscape contains low-curvature directions that span the whole layer chain, and a roughly 28-fold larger gradient budget leaves the test accuracy of 256-layer networks unchanged. A residual PC chain, however, supplies its own solver for these directions. Linearizing the predictions at the feedforward state yields a batch-shared Gauss–Newton preconditioner whose inverse is applied exactly by one backward and one forward recurrence over the layers, plus a small cached output-space correction. Cached Boundary Transport (CBT) uses this solve for the chain-wide component and a short gradient-descent refinement for the remaining local error; an energy acceptance criterion and a restricted Gauss–Newton fallback handle nonlinear mismatch. From 16 to 256 layers, CBT keeps inference cost nearly constant in gradient evaluations while first-order and Newton-type solvers grow with depth. On trained networks it brings the parameter gradient substantially closer to that of an exact second-order oracle, and in end-to-end training it is more accurate than depth-proportional inference at every tested depth with up to $17\times$ fewer FLOPs. These results point to chain geometry, not the step budget, as the bottleneck of deep residual PC inference.

1 INTRODUCTION

Predictive coding (PC) trains a hierarchy through local prediction errors (Rao & Ballard, 1999). Let $z_0 = x$ be the input, $z = (z_1, \dots, z_K)$ the adjustable hidden activities, and f_ℓ the prediction from one layer to the next. For a supervised target y, a common PC energy is

$$\mathcal{F}(z; W) = \frac{1}{2} \sum_{\ell=0}^{K-1} \|z_{\ell+1} - f_\ell(z_\ell; W_\ell)\|_2^2 + \frac{1}{2} \|y - f_K(z_K; W_K)\|_2^2. \tag{1}$$

Training alternates two operations. It first infers activities $z^\star = \arg\min_z \mathcal{F}(z; W)$ for fixed weights, then updates W from the prediction errors at the inferred activities. This inner optimization is *activity inference*. It is repeated before every parameter update, and the inferred activities are the learning signal: PC approximates or implements backpropagation-like credit assignment with local parameter updates when this inner problem is solved well (Whittington & Bogacz, 2017; Salvatori et al., 2022).

Depth turns this inner optimization into a geometric, not merely iterative, problem. As μPC showed, the activity-inference landscape becomes increasingly ill-conditioned with width and particularly depth, and μPC itself does not remove this depth dependence (Innocenti et al., 2026). In a local quadratic mode with curvature λ_i, gradient descent (GD) contracts error by $1 - \eta\lambda_i$: high curvature limits the stable step size while low curvature decays slowly. In a deep PC chain, the slow directions coordinate activities along many successive prediction Jacobians. A scalar residual model makes the consequence explicit: its condition number grows as K^2 (Appendix A.1). The same tension appears in learning. In our width-64, 256-layer residual network, increasing the inference budget from 9 to 256 GD steps changes final test accuracy from 87.64% to 87.49%, although it advances the training plateau from epoch 45 to epoch 18. More steps reach the same accuracy sooner; they do not raise it. The bottleneck is the geometry of the chain, not the number of iterations.

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

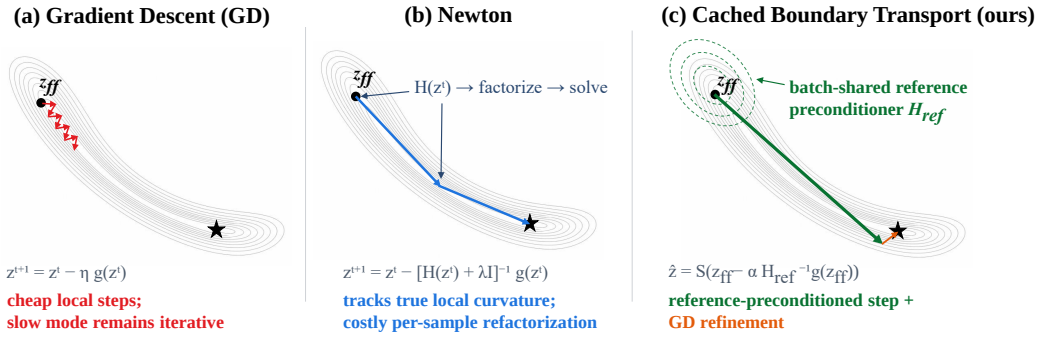


Figure 1: **Three strategies for ill-conditioned activity inference.** GD makes inexpensive steps but resolves the slow mode iteratively. The Newton alternative uses current curvature but requires repeated sample-specific factorization. CBT instead applies a batch-shared reference preconditioner, refines the proposal with GD, and applies the energy acceptance criterion. A proposal that fails the criterion uses a restricted Gauss–Newton fallback. Here $g(z) = \nabla_z \mathcal{F}(z)$, $H(z) = \nabla_z^2 \mathcal{F}(z)$, and $\mathcal{S}$ denotes refinement and the energy acceptance criterion specified in Section 3.3.

Second-order curvature can rescale these directions, but the direct remedy is costly. A regularized exact-Hessian Newton step uses a sample-specific matrix that changes after every accepted update and contains residual-weighted second derivatives. Even exploiting its block-tridiagonal structure requires repeated dense-width factorizations. Under our complete-update cost model, one such iteration costs $2.0\times$ one CBT update at width 16 and $31.8\times$ at width 256 (Appendix A.5). Exact-Hessian Newton is therefore a valuable oracle, and we use it as one, but an expensive inner loop for training.

Our key observation is that a residual PC chain already contains a solver for its depth-coherent component. At the feedforward state, we fix one set of prediction Jacobians shared by the minibatch. The resulting Gauss–Newton reference matrix couples every hidden layer, yet its inverse is applied by one backward recurrence, one forward recurrence, and a cached output-dimensional correction. Cached Boundary Transport (CBT) spends this solve on the coordinated component and a few ordinary GD steps on the remaining nonlinear error. An energy criterion accepts the proposal; when it does not, a Gauss–Newton step with current Jacobians on the last few layers takes over. Inference becomes a two-scale problem: solve the chain, then refine locally (Figure 1).

Three levels of evidence support this account. First, on exact linear chains, a single chain solve reaches the analytic minimum in 3.33–3.49 gradient equivalents across 16–256 layers, whereas oracle-tuned GD needs 342 to more than 20,000 to reach a 10^{-6} relative energy gap. Second, nonlinear epoch-zero traces show the division of labor: at 256 layers, the same short GD refinement removes 20.6 percentage points of feedforward energy after the chain solve and 0.7% without it. Third, on trained 128-layer checkpoints, CBT produces parameter gradients substantially closer to an exact-Hessian oracle than matched-cost GD or heavy ball (cosine 0.58 versus 0.34–0.35), and end-to-end training shows the corresponding accuracy advantage across depth.

CBT builds on classical linear algebra: triangular solves and the Woodbury identity. The contribution is to recognize this structure in nonlinear PC inference and turn it into a batch-shared training solver. The structure is carried by residual connections, which keep products of layer Jacobians from contracting across depth; removing them removes the advantage, which confirms the mechanism (Appendix I).

Our contributions are:

- We expose a chain-wide component of deep residual PC inference and derive a positive-definite, batch-shared Gauss–Newton reference solve whose inverse requires two layerwise recurrences and a cached output correction.
- We turn that solve into a two-scale nonlinear method: a chain solve, a short GD refinement, an energy criterion, and a restricted Gauss–Newton fallback. Both search directions are strict descent directions whenever the relevant gradient is nonzero (Section 3).

- We give an account of what limits deep residual PC inference. From exact linear chains to trained 256-layer networks, the evidence indicates a depth-coherent component that additional gradient steps do not remove; removing it improves the parameter gradient used for learning, and the mechanism depends on residual connections that keep the Jacobian chain from contracting.

2 Background and Related Work

Predictive coding and its inner optimization. PC networks alternate activity inference with parameter updates. Under suitable schedules and assumptions, the resulting local updates approach backpropagation (Whittington & Bogacz, 2017), while computational-graph constructions can realize exact reverse differentiation (Salvatori et al., 2022); Millidge et al. (2022) review this landscape. The inference dynamics themselves need not converge outside restricted small-weight regimes (Frieder & Lukasiewicz, 2022). Alonso et al. (2024) introduce sequential inference, which updates activities layer by layer from the output backward to reduce error-propagation delay and thereby reduce the number of inference iterations; μPC stabilizes deep residual PC through parameterization (Innocenti et al., 2026); and Error Optimization changes the inferred variables to address signal decay (Goemaere et al., 2025). CBT instead keeps the original free activities and directly preconditions their chain-wide coupling.

Structured and implicit solvers. Deep equilibrium models replace an explicit-depth forward computation with a root solve and use implicit differentiation at equilibrium (Bai et al., 2019). CBT instead solves the activities of an explicit PC chain; its distinguishing object is the block-bidiagonal residual Jacobian. Residual connections (He et al., 2016) also determine whether this long-range Jacobian path remains useful in our experiments.

Chain-structured numerical optimization. Preconditioning and line search are standard ways to reshape a problem and globalize a finite proposal (Nocedal & Wright, 2006). Forward–backward recursions and block-tridiagonal solves are likewise classical for chain-structured least squares, especially linear-Gaussian smoothing (Rauch et al., 1965; Aravkin et al., 2017), and low-rank inverse updates follow from the Woodbury identity (Woodbury, 1950). CBT's distinguishing object is their composition for PC: a batch-shared reference preconditioner, an output-dimensional correction, a reusable cache, and an energy criterion for the nonlinear objective. Unlike classical smoothing, the variables are activities in a learned nonlinear network, so the reference matrix serves as a preconditioner for a nonlinear energy.

PC as a digital training algorithm. As in μPC (Innocenti et al., 2026) and JPC (Innocenti et al., 2024), we treat PC as a digital training algorithm. Parameter updates remain local to prediction errors, and the two recurrences exchange information only between adjacent layers; inference, in turn, coordinates the full chain through these messages and a cached output-space correction.

3 Cached Boundary Transport

CBT is a two-scale solver. It first applies a batch-shared Gauss–Newton reference inverse to the true activity gradient, targeting a component coherent across the layer chain. It then takes m GD refinements for the remaining nonlinear error. An energy criterion accepts the complete proposal; otherwise a Gauss–Newton step built from the current, sample-specific Jacobians and restricted to the last few layers takes over. Unless stated otherwise, "CBT" denotes this complete procedure. In the name, *boundary* is the supervised output residual, and *transport* is the pair of recurrences used to apply the reference inverse.

3.1 Objective and a shared SPD reference preconditioner

Let $z = (z_1, \dots, z_K)$ denote the free hidden activities for one example, with $z_\ell \in \mathbb{R}^N$. Predictions are $f_0(x)$, $f_\ell(z_\ell)$ for $1 \le \ell < K$, and an output map $f_K(z_K)$. We define

$$e_1 = z_1 - f_0(x), \qquad e_{\ell+1} = z_{\ell+1} - f_\ell(z_\ell), \tag{2}$$

$$e_{\text{out}} = y - f_K(z_K), \qquad \mathcal{F}(z; W) = \frac{1}{2}\sum_{\ell=1}^{K} \|e_\ell\|_2^2 + \frac{1}{2}\|e_{\text{out}}\|_2^2. \tag{3}$$

Activity inference minimizes $\mathcal{F}$ for fixed parameters W. The parameter gradient is then evaluated while holding the inferred activities fixed:

$$\hat{z} \leftarrow \text{Infer}(\mathcal{F}, z_{\text{ff}}), \qquad W^+ \leftarrow W - \rho\, \nabla_W \mathcal{F}(\hat{z}; W). \tag{4}$$

The inferred activities therefore control both the inner cost and the prediction errors used for learning.

At the feedforward state z_{ff}, all hidden prediction errors vanish. The implementation constructs its shared Jacobians explicitly as follows. For a minibatch of size B, let $\bar{a}_\ell = B^{-1}\sum_{b=1}^{B} \phi'(z_{\ell,b}^{\text{ff}})$. In the dense residual models used here, $f_\ell(z) = z + s_\ell W_\ell \phi(z)$, and hence

$$J_\ell = I + s_\ell W_\ell \,\text{diag}(\bar{a}_\ell), \qquad C = s_K W_K \,\text{diag}(\bar{a}_K). \tag{5}$$

Thus the activation derivative is averaged before it is inserted into the Jacobian. For this architecture, Equation 5 is also the batch mean of the sample Jacobians at z_{ff}, and it serves as a fixed reference for all subsequent activities of the minibatch.

The difficult directions motivate the following construction. A low-energy perturbation approximately satisfies $\delta z_{\ell+1} \approx J_\ell \delta z_\ell$, so correcting one layer requires a coherent change along the chain. Scalar-step GD does not perform this transport explicitly. Define the block-bidiagonal matrix L by

$$(Ld)_1 = d_1, \qquad (Ld)_{\ell+1} = d_{\ell+1} - J_\ell d_\ell. \tag{6}$$

Because every diagonal block is the identity, L is nonsingular regardless of depth. If S_K selects the last activity block, the shared reference residual Jacobian is the vertical concatenation of L and $-CS_K$. Its Gauss–Newton matrix is

$$H_{\text{ref}} = L^\top L + S_K^\top C^\top C S_K. \tag{7}$$

It is positive definite because $L^\top L \succ 0$.

3.2 Cached application of the reference preconditioner

Ignoring the output term for a moment, the hidden-chain inverse is $(L^\top L)^{-1} = L^{-1}L^{-\top}$. For a right-hand side g, the vector $u = L^{-1}L^{-\top}g$ follows from two recurrences:

$$q_K = g_K, \qquad q_\ell = g_\ell + J_\ell^\top q_{\ell+1}, \tag{8}$$

$$u_1 = q_1, \qquad u_{\ell+1} = q_{\ell+1} + J_\ell u_\ell. \tag{9}$$

Equation 8 is the backward transport in the method name, and Equation 9 is the forward transport.

Supervision adds a low-rank correction. Let

$$R = CS_K L^{-1}. \tag{10}$$

Then $H_{\text{ref}} = L^\top(I + R^\top R)L$, and the Woodbury identity gives

$$H_{\text{ref}}^{-1} = L^{-1}\left[I - R^\top (I + RR^\top)^{-1} R\right] L^{-\top}. \tag{11}$$

The dense system $I + RR^\top$ has size $O \times O$, independent of depth and hidden width.

A direct evaluation of Equation 11 would require another forward recurrence for every minibatch gradient. We instead precompute the response to the O output-space basis vectors:

$$U = L^{-1}R^\top, \qquad M = I + CS_K U. \tag{12}$$

For each minibatch activity gradient g, CBT computes

$$q = L^{-\top} g, \qquad u = L^{-1} q, \qquad v = M^{-1} C S_K u, \qquad d = u - U v. \tag{13}$$

Substitution of Equation 12 into Equation 11 shows that $d = H_{\mathrm{ref}}^{-1} g$ up to floating-point error. No full $KN \times KN$ matrix is formed. After cache construction, one application comprises the two recurrences above, an $O \times O$ solve, and the cached output correction Uv.

Crucially, CBT approximates curvature, not the activity gradient. For an individual example let $g_b = \nabla_z \mathcal{F}_b(z)$ and take the step $s_b = -H_{\mathrm{ref}}^{-1} g_b$. Since $H_{\mathrm{ref}} \succ 0$, whenever $g_b \neq 0$,

$$g_b^\top s_b = -g_b^\top H_{\mathrm{ref}}^{-1} g_b < 0. \tag{14}$$

The reference-preconditioned step is therefore a strict descent direction for the true nonlinear energy whenever $g_b \neq 0$. Section 3.3 selects its finite length by evaluating that energy directly. Appendix A.4 gives the cache recurrence and operation count.

Cost. For dense layers of width N, minibatch size B, and output size O, applying the reference inverse costs $4B(K-1)N^2$ FLOPs at leading order, the same order as one activity-gradient evaluation: both traverse the chain once in each direction with B right-hand sides. The cache repeats this traversal with O output-space right-hand sides once per minibatch, and it stores only the J_ℓ, the blocks U_ℓ, and an $O \times O$ Cholesky factor, i.e., $O(KN^2 + KNO)$ memory. The reference step therefore costs a constant number of gradient evaluations at any depth (2.8–3.7 observed from $K = 16$ to 256), whereas T GD steps cost T; a depth-proportional budget $T = K$ makes the activity cost quadratic in depth.

3.3 GD REFINEMENT, ENERGY ACCEPTANCE CRITERION, AND FALLBACK

The reference-preconditioned step targets modes that span the full chain. We follow it by m ordinary GD steps to reduce the nonlinear residual at the proposed activities. Define

$$\mathrm{GD}_\eta^m(z) = \underbrace{\mathrm{GD}_\eta \circ \cdots \circ \mathrm{GD}_\eta}_{m \text{ steps}}(z). \tag{15}$$

For a primary scale α_0 and a smaller scale α_1, the ordered candidates are

$$z^{(0)} = \mathrm{GD}_\eta^m(z_0 - \alpha_0 d), \quad z^{(1)} = \mathrm{GD}_\eta^m(z_0 - \alpha_1 d), \quad z^{(2)} = \mathrm{GD}_\eta^m(z_0), \quad z^{(3)} = z_0, \tag{16}$$

where $z_0 = z_{\mathrm{ff}}$. The primary proposal $z^{(0)}$ therefore uses one reference-preconditioned step and the same number of GD refinements at every depth. Each GD step still visits all K layers, so raw FLOPs grow with depth. The fixed quantity is the number of refinement steps, not their raw cost.

Equation 14 fixes the direction; the fixed scales in Equation 16 are validated on the true energy, which captures sample-to-sample Jacobian variation, Jacobian drift, and residual-weighted activation curvature. A candidate for sample b is retained when it satisfies the *energy acceptance criterion*

$$\mathcal{A}_\varepsilon(z_b) = \mathbf{1}[z_b \text{ is finite and } \mathcal{F}_b(z_b) \leq \mathcal{F}_b(z_{0,b}) + \varepsilon \max\{1, |\mathcal{F}_b(z_{0,b})|\}]\,. \tag{17}$$

For each sample, let p_b be the first candidate satisfying the energy acceptance criterion in the order $z_b^{(0)}, z_b^{(1)}, z_b^{(2)}, z_{0,b}$. The last candidate always satisfies the criterion, so this selection is well defined.

If the primary proposal fails the energy acceptance criterion for any sample, the implementation executes one batch-level fallback branch. Let S_h contain the last h activity blocks and let E_{S_h} inject a vector on those blocks. At p_b, define the current sample-specific Gauss–Newton matrix $G_b(p_b) = \mathcal{J}_b(p_b)^\top \mathcal{J}_b(p_b)$ and the restricted direction

$$\Delta_b = -E_{S_h} \left(E_{S_h}^\top G_b(p_b) E_{S_h} \right)^{-1} E_{S_h}^\top \nabla_z \mathcal{F}_b(p_b). \tag{18}$$

The hidden part of $\mathcal{J}_b(p_b)$ is unit block-bidiagonal, hence has full column rank. Therefore the restricted matrix is positive definite: Δ_b is zero when the projected gradient is zero and is otherwise a strict descent direction (Appendix A).

A scale γ is selected from a finite set Γ by

$$\mathcal{F}_b(p_b + \gamma \Delta_b) \leq \mathcal{F}_b(p_b) + c\gamma \nabla \mathcal{F}_b(p_b)^\top \Delta_b. \tag{19}$$

Armijo acceptance decreases energy from p_b and therefore preserves the energy acceptance criterion; the implementation also rechecks the criterion after restoring the restricted blocks and otherwise retains p_b. Every returned activity therefore has energy no higher than the feedforward state, up to the tolerance ε. Algorithm 1 summarizes the complete procedure.

Algorithm 1 Cached Boundary Transport activity inference

Require: Network W, minibatch (x, y); $\alpha_0 > \alpha_1 > 0$, $m, \eta, h, \Gamma, c, \varepsilon$

1: $z_0 \leftarrow z_{\text{ff}}(x)$; build $\{J_\ell, U_\ell, M\}$ at z_0
2: $d \leftarrow H_{\text{ref}}^{-1} \nabla_z \mathcal{F}(z_0)$ using Equation 13
3: Evaluate the candidates in Equation 16 in order and only as needed; for each sample, retain the first one satisfying the energy acceptance criterion in Equation 17
4: **if** the primary candidate $z^{(0)}$ fails the energy acceptance criterion for any sample **then**
5: Form the restricted Gauss–Newton direction in Equation 18
6: Retain the first trial with $\gamma \in \Gamma$ satisfying Equation 19 and the energy acceptance criterion; otherwise retain the previous candidate satisfying the criterion
7: **end if**
8: **return** inferred activities $\hat{z}$

4 Experiments

The experiments follow the proposed mechanism into learning. We ask: does the chain solve reach the component that first-order iterations reach only slowly, first when the reference is exact and then in nonlinear inference (Section 4.1); does the resulting activity state improve the parameter gradient (Section 4.2); does that improvement alter depth and width scaling in training (Section 4.3); and what do the nonlinear safeguards contribute (Section 4.4)? Unless stated otherwise, results are means with sample standard deviations over three paired seeds, and we report paired differences.

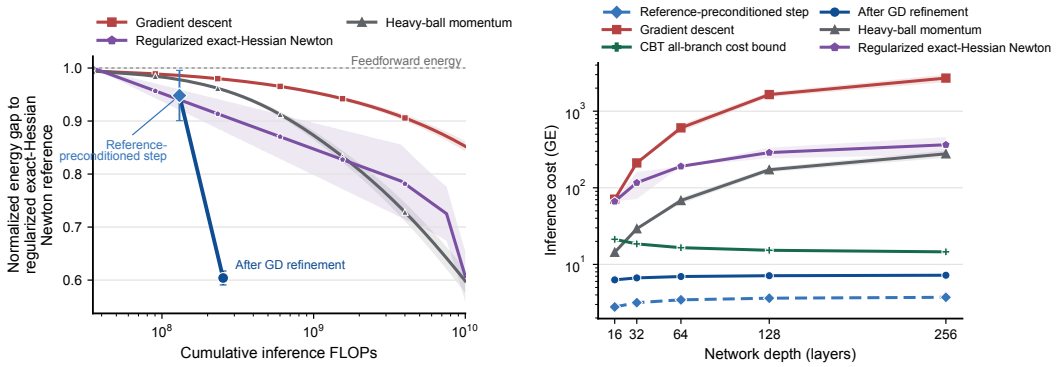
Training experiments use the JPC residual model (Innocenti et al., 2024) with μPC depth scaling (Innocenti et al., 2026), float64, and Adam at 10^{-3} (Kingma & Ba, 2014). Paired methods share initialization, splits, and minibatch order; validation selects one complete epoch before a single test. The fixed CBT configuration uses $\alpha_0 = 0.2$, $\alpha_1 = 0.02$, $m = 3$, $\eta = 0.3$, $h = 16$, $\Gamma = \{1, 1/2, 1/4, 1/8, 1/16\}$, $c = 10^{-4}$, and $\varepsilon = 10^{-10}$ throughout. These constants are set once and reused without retuning across depths 16–256, widths 64–256, three datasets, and two activations; m and h are ordinary solver hyperparameters rather than depth-dependent budgets. Appendix B gives the full protocol and accounting rules.

We first count total FLOPs, including setup, executed solver branches, parameter gradients, warm-up, validation, and final testing. We report normalized compute in gradient equivalents (GE) by dividing this total by the FLOPs of one depth- and width-matched activity-gradient evaluation. FLOPs are the primary efficiency measure; A100 time is reported relative to baselines on the same device. We compare with calibrated compute-matched GD and heavy ball (GD-CM/HB-CM), GD with one step per hidden layer (GD-K, the setting used in the μPC depth experiments), a 128-step GD width baseline, and regularized exact-Hessian Newton used only as a mechanism oracle. Step budgets are fixed before the reported training comparisons.

4.1 Does the chain solve reach the mode that iterations miss?

We begin where the reference is exact. On width-8 linear residual chains with noncommuting orthogonal layers, a single unit-step chain solve without refinement reaches the analytic minimum across $K = 16$–256 and three seeds: the largest relative optimality gap is 8.42×10^{-19}, at a cost of 3.33–3.49 GE. To reach even a relative energy gap of 10^{-6}, oracle-tuned GD grows from 342 GE at $K = 16$ to more than 20,000 GE at $K = 256$; heavy ball grows from 45 to 592 GE. The chain solve reaches the same endpoint as a shared exact-Newton implementation, which costs 1.29–$1.33\times$ as much. Measured against the exact minimizer, the chain solve is exact at nearly constant GE across depth (Appendix A.2).

We next isolate nonlinear inference on epoch-zero residual μPC MLPs with $N = 64$, batch size eight, five initializations, and $K \in \{16, 32, 64, 128, 256\}$. We measure the reference-preconditioned step, the iterate after GD refinement, and the final CBT iterate. GD, HB, and per-sample regularized exact-Hessian Newton start from the same feedforward activities and stop when they first reach the final CBT energy for that run. The cost includes gradients, Hessians, factorizations, solves, rejected trials, evaluations of the energy acceptance criterion, cache construction, and every executed branch. Appendix E reports the stage allocation and raw FLOPs.



(a) Stage-resolved energy at $K = 256$. **(b)** Compute to the observed final CBT energy.

Figure 2: **A chain solve followed by short GD refinement avoids depth-growing iteration.** Mean $\pm$ s.d. over five network seeds. (a) Energy is normalized to one at feedforward and zero at the converged regularized exact-Hessian Newton reference; curves are shown only through 10^{10} inference FLOPs. (b) GD, HB, and regularized exact-Hessian Newton stop when they first match the per-seed final CBT energy. The all-branch bound charges every possible fallback. The observed final iterate equals the iterate after GD refinement because no fallback is executed at epoch zero.

At $K = 256$, the reference-preconditioned step costs 3.721 GE and removes $3.09\% \pm 2.80\%$ of the feedforward energy. The m GD refinements then remove another 20.63 percentage points, for $23.72\% \pm 1.53\%$ total reduction at 7.221 GE (Figure 2a). The same refinement started from feedforward remove $0.730\% \pm 0.076\%$, and at the full 7-GE budget GD and HB remove $1.24\% \pm 0.13\%$ and $2.40\% \pm 0.25\%$. The chain solve removes little energy itself, as expected of a low-curvature component, but it changes which error the short GD refinement can reach: a 28-fold increase in what the same refinement removes. The reference step accounts for 44–51% of observed inference FLOPs: the step that removes the least energy is the largest cost and makes the rest effective (Appendix E). The fallback is dormant at initialization in all 25 runs; Section 4.4 evaluates it on trained states.

The cost of the final CBT iterate changes from 6.284 GE at $K = 16$ to 7.221 GE at $K = 256$. In contrast, the cost to reach the same per-run energy increases with depth for all three comparators. At $K = 256$, GD requires 2703.4 ± 307.2 GE, HB 277.4 ± 30.7 GE, and regularized exact-Hessian Newton 364.6 ± 95.6 GE (Figure 2b). Raw FLOPs grow linearly with depth for every method because each gradient visits every layer (Appendix E); measured in gradient evaluations, CBT's cost stays bounded while that of every comparator grows.

4.2 Does better inference change the learning signal?

We freeze trained width-64, 128-layer tanh checkpoints from three seeds and evaluate one deterministic 16-example validation batch per seed. A converged per-sample regularized exact-Hessian Newton solve defines an oracle activity state. For an inferred state a, its oracle energy quality is $q_E(a) = [\mathcal{F}(z_{\text{ff}}) - \mathcal{F}(a)]/[\mathcal{F}(z_{\text{ff}}) - \mathcal{F}(z_{\text{oracle}})]$. GD and HB receive the complete-update FLOP budget of CBT, and all parameter gradients are evaluated while holding the corresponding activities fixed.

CBT captures $48.7\% \pm 9.3\%$ of the oracle energy reduction, compared with $5.6\% \pm 1.6\%$ for GD and $12.8\% \pm 4.7\%$ for HB. More importantly, the cosine between the resulting parameter gradient and the oracle gradient is 0.34 at feedforward, 0.34 after matched-cost GD, 0.35 after HB, and 0.58 after CBT. Matched-cost first-order inference leaves the learning signal where feedforward left it; CBT raises its alignment with the oracle from 0.34 to 0.58. The first exact-Hessian Newton prefix that reaches CBT's energy quality costs 25.8–30.1$\times$ as many complete-update FLOPs, and at that matched energy quality its gradient cosine is 0.54 rather than 0.58. Energy reduction alone does not determine gradient fidelity; the solver advantage reaches the signal used by the outer optimization, not only the inner objective. Appendix C gives the protocol and per-seed endpoints.

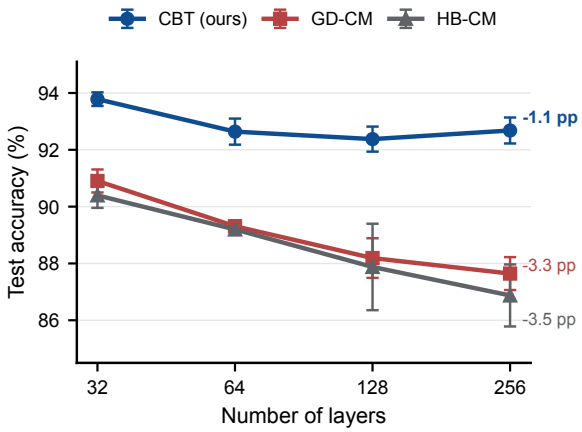


Figure 3: **Depth scaling under worst-case-calibrated inference budgets.** Test accuracy is mean ± s.d. over three seeds. End annotations show the accuracy change from 32 to 256 layers for each method.

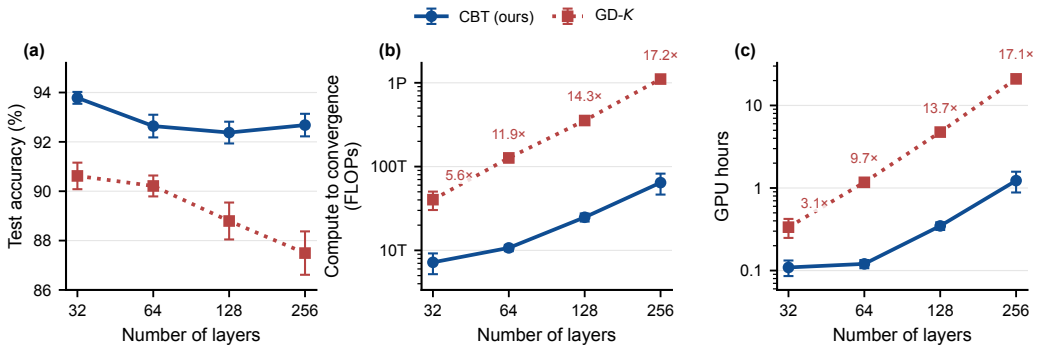


Figure 4: **Practical depth-proportional inference.** GD-K uses as many activity steps as layers. Compute is total dominant FLOPs through the selected validation epoch; time is baseline-relative complete-update time on the same A100. Cost-panel labels report GD-K/CBT ratios.

4.3 Does the learning advantage persist with scale?

Compute-matched depth scaling. The canonical task is MNIST with tanh residual MLPs, width $N = 64$, batch size 64, and $K \in \{32, 64, 128, 256\}$. GD-CM and HB-CM use integer step budgets chosen on a separate seed against the worst-case CBT cost, so both baselines receive at least the compute CBT actually uses at every depth.

Across 32–256 layers, CBT's mean test accuracy remains between 92.4% and 93.8%, whereas GD-CM falls from 90.9% to 87.6%. The mean CBT margin over GD-CM grows from 2.88 to 5.04 points; the corresponding HB-CM margins are 3.39 and 5.81 points (Figure 3). Across the four depths, GD-CM uses 2.25–3.84× the total GE of CBT and HB-CM uses 1.80–3.61×. Because the budgets are calibrated to CBT's worst-case update, the baselines receive 1.10–2.23 times CBT's realised per-update compute at $K \leq 64$ and match it within 0.4% at $K = 256$, where the margin is largest (Appendix B, Table 2). The advantage does not depend on the plateau rule: GD-CM needs 23–47× the GE of CBT to reach stable 85% validation accuracy (Appendix D).

Practical depth-proportional inference. Following the depth experiments of μPC (Innocenti et al., 2026), which use one inference step per hidden layer, we also evaluate a depth-proportional baseline, GD-K, with three paired seeds at every depth, using our shared training setup.

Across $K = 32$–256, CBT raises test accuracy over GD-K by 2.43–5.19 points (Figure 4a) while reducing FLOPs 5.6–17.2× and measured complete-update time 3.1–17.1× (Figures 4b–c). The

difference grows because one GD step traverses the chain once: allocating K steps to a K-layer model makes this part of the activity cost quadratic in depth. At $K = 256$, extending GD from the nine-step compute-matched budget to 256 steps advances the training plateau from 45 ± 5 to 18 ± 2 epochs, but leaves final test accuracy at $87.64\% \pm 0.58\%$ versus $87.49\% \pm 0.88\%$. A larger budget reaches the same accuracy sooner; CBT reaches a higher one.

Width scaling. At fixed depth $K = 128$, we vary $N \in \{64, 128, 256\}$ without changing the solver configuration. We compare CBT with GD-CM and with the higher-budget GD-128 baseline.

Width narrows the accuracy gap but not the compute gap. CBT is more accurate than GD-CM at every width, by 4.19 points at $N = 64$ and 0.66 points at $N = 256$, while GD-CM uses 2.3–2.8$\times$ its FLOPs (Figure 10, Appendix F). The 128-step GD baseline, at 14.3–19.9$\times$ the FLOPs of CBT, trails it by 3.58 points at $N = 64$, matches it within 0.04 points at $N = 128$, and exceeds it by 0.84 points at $N = 256$ for 17.7$\times$ the FLOPs. Wider layers also make the shared reference more faithful: the fraction of training updates that execute the fallback falls from 93% at $N = 64$ to 53% and 25% at $N = 128$ and 256, so CBT's realised per-update cost decreases with width while that of GD-CM stays fixed. CBT thus holds the efficient end of the accuracy–compute frontier across width.

Dataset transfer. Without retuning, CBT exceeds GD-CM by 3.38 points on Fashion-MNIST and 3.18 points on a flattened CIFAR-10 residual MLP, with every paired CIFAR-10 difference positive. On CIFAR-10, GD-CM and GD-K use 2.68$\times$ and 7.43$\times$ the total GE of CBT (Appendix H.1).

Activation transfer. With SiLU instead of tanh at $K = 128$, the same configuration reaches $93.37\% \pm 1.07\%$ test accuracy (tanh: $92.38\% \pm 0.44\%$), and every trajectory plateaus before the epoch cap (Appendix H).

4.4 What do the nonlinear safeguards contribute?

The two safeguards address different errors. The output correction makes the shared proposal more faithful to the supervised objective: it lowers the fraction of raw proposals that increase energy by 12.0 and 13.5 points at $K = 64$ and 128. The restricted Gauss–Newton fallback protects learning where nonlinear mismatch is large: removing it lowers SiLU validation accuracy by 26.9 points, whereas tanh is essentially unaffected (0.05 points) (Figure 11, Appendix G). The fallback runs only when needed: over the full validation set of the $K = 64$ tanh checkpoints, the energy trigger skips it on 54 of 234 batches, and every batch returns activities that satisfy the energy acceptance criterion.

5 Discussion and Conclusion

How much energy a step removes is not the right measure of its value in deep PC inference. At 256 layers, the chain solve removes 3.1% of the feedforward energy, yet the subsequent GD refinement then removes 28 times more energy than it does from feedforward. Exact linear controls, trained-checkpoint oracle comparisons, and end-to-end scaling all point to the same picture: a low-curvature, depth-coherent component carries little energy, resists additional gradient steps, and governs access to a useful learning signal. The residual chain provides an exact solve for it.

This changes where inference compute should go. Deep PC protocols commonly scale the number of inference steps with depth. In our experiments, that budget grows quadratically in raw cost and reaches its accuracy plateau sooner, but does not raise the plateau; a fixed structured solve does, at a fraction of the cost. In deep residual PC, spending inference compute on the structure of the chain is more effective than spending it on more iterations.

Residual connections are what make this structure usable. With them, the end-to-end Jacobian gain stays nearly constant with depth; without them it contracts and the solver ordering reverses (Appendix I), which pins the mechanism to the preserved Jacobian path. The present evidence covers dense residual PC networks on image benchmarks with three training seeds, and FLOPs carry the efficiency claim. Convolutional and larger-scale PC models remain untested and are the natural next step for the chain solve. Deep predictive-coding inference need not be an ever longer sequence of gradient steps: when residual connections preserve the Jacobian chain, the chain itself solves the component that those steps reach only slowly.

AI USE STATEMENT

We used generative AI tools to assist with writing and editing, literature search and discovery, and code development for training, evaluation, and visualization. We did not use generative AI for research ideation or to generate synthetic data or results. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the work.

ETHICS STATEMENT

This study uses established public image benchmarks and does not involve human participants, private data, or newly collected personal information. Its main direct societal cost is the computation used for repeated training and evaluation; we therefore report FLOPs and describe the hardware measurements and their limitations.

REPRODUCIBILITY STATEMENT

Algorithm 1 and Appendix B specify the solver, stopping rules, seeds, numerical precision, baselines, and FLOP accounting. We will release the code, configuration files, evaluation scripts and trained checkpoints.

REFERENCES

Nicholas Alonso, Jeffrey Krichmar, and Emre Neftci. Understanding and improving optimization in predictive coding networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10812–10820, 2024.

Aleksandr Aravkin, James V Burke, Lennart Ljung, Aurelie Lozano, and Gianluigi Pillonetto. Generalized Kalman smoothing: Modeling and algorithms. *Automatica*, 86:63–86, 2017.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Simon Frieder and Thomas Lukasiewicz. (Non-)Convergence results for predictive coding networks. In *International Conference on Machine Learning*, pp. 6793–6810. PMLR, 2022.

Cédric Goemaere, Gaspard Oliviers, Rafal Bogacz, and Thomas Demeester. Error optimization: Overcoming exponential signal decay in deep predictive coding networks. *arXiv e-prints*, pp. arXiv–2505, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Francesco Innocenti, Paul Kinghorn, Will Yun-Farmbrough, Miguel De Llanza Varona, Ryan Singh, and Christopher L Buckley. JPC: Flexible inference for predictive coding networks in JAX. *arXiv preprint arXiv:2412.03676*, 2024.

Francesco Innocenti, El Mehdi Achour, and Christopher L Buckley. μPC: Scaling predictive coding to 100+ layer networks. *Advances in Neural Information Processing Systems*, 38:87414–87455, 2026.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Beren Millidge, Tommaso Salvatori, Yuhang Song, Rafal Bogacz, and Thomas Lukasiewicz. Predictive coding: Towards a future of deep learning beyond backpropagation? *arXiv preprint arXiv:2202.09467*, 2022.

J Nocedal and S Wright. Numerical optimization, 2006.

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999.

Herbert E Rauch, Fong Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Tommaso Salvatori, Yuhang Song, Zhenghua Xu, Thomas Lukasiewicz, and Rafal Bogacz. Reverse differentiation via predictive coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8150–8158, 2022.

James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29(5): 1229–1262, 2017.

Max A Woodbury. *Inverting modified matrices*. Department of Statistics, Princeton University, 1950.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A Derivations and Cost

A.1 A scalar slow mode

The simplest chain already shows why depth creates a difficult activity direction. Consider scalar activities with identity predictions,

$$\mathcal{F}(z) = \frac{1}{2}(z_1 - x)^2 + \frac{1}{2}\sum_{\ell=1}^{K-1}(z_{\ell+1} - z_\ell)^2 + \frac{1}{2}(y - z_K)^2. \tag{20}$$

Its activity Hessian is the $K \times K$ tridiagonal matrix with 2 on the diagonal and -1 on the two adjacent diagonals. Its eigenvalues are

$$\lambda_j = 2 - 2\cos\left(\frac{j\pi}{K+1}\right), \qquad j = 1, \dots, K. \tag{21}$$

Consequently, $\lambda_{\min} = \Theta(K^{-2})$, $\lambda_{\max} = \Theta(1)$, and the condition number is $\Theta(K^2)$. With a stable depth-independent GD step, the slowest error component contracts as $1 - \Theta(K^{-2})$ and therefore needs $\Theta(K^2 \log(1/\delta))$ steps to shrink by a factor δ. Nonlinear prediction Jacobians change the modes and their scaling, but the coherent across-depth direction persists; it is the component the chain solve targets.

A.2 Exact linear-chain control

The exactness control uses width-8 linear residual chains with noncommuting orthogonal layer maps, depths $K \in \{16, 32, 64, 128, 256\}$, and three seeds. Because the residual Jacobian is constant, the reference matrix equals the true Gauss–Newton matrix. One chain solve reaches the analytic least-squares minimum with maximum relative gap 8.42×10^{-19} and costs 3.33–3.49 GE across depth. Oracle-tuned first-order methods instead require

K	16	32	64	128	256
GD (GE)	342	1,217	4,429	16,281	$>$ 20,000
HB (GE)	45	85	160	306	592

to reach a relative energy gap of 10^{-6}; the last GD cell is right-censored at 20,000 iterations. The chain solve here is a single unit-step application of Equation 13 without GD refinement, so it isolates the reference inverse from the nonlinear safeguards. A shared exact-Newton implementation also reaches that endpoint but uses 1.29–$1.33\times$ the chain-solve FLOPs. This experiment removes nonlinear Jacobian drift, stopping-target choice, and training dynamics from the comparison.

A.3 Reference preconditioner

Stack the linearized hidden residuals. At z_{ff},

$$\begin{bmatrix} \delta e_1 \\ \delta e_2 \\ \vdots \\ \delta e_K \end{bmatrix} = \underbrace{\begin{bmatrix} I & 0 & \cdots & 0 \\ -J_1 & I & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & -J_{K-1} & I \end{bmatrix}}_{L} \begin{bmatrix} \delta z_1 \\ \delta z_2 \\ \vdots \\ \delta z_K \end{bmatrix}. \tag{22}$$

The output residual contributes $-CS_K\delta z$. Signs disappear in the normal matrix, which yields $H_{\mathrm{ref}} = L^\top L + S_K^\top C^\top C S_K$.

Since L is triangular with identity diagonal blocks,

$$H_{\mathrm{ref}} = L^\top \left[I + L^{-\top} S_K^\top C^\top C S_K L^{-1} \right] L = L^\top (I + R^\top R) L. \tag{23}$$

Applying Woodbury to $(I + R^\top R)^{-1}$ proves Equation 11. Defining $U = L^{-1}R^\top$ gives $RR^\top = CS_KU$. Therefore

$$H_{\mathrm{ref}}^{-1} g = L^{-1}L^{-\top} g - L^{-1}R^\top (I + RR^\top)^{-1} R L^{-\top} g \tag{24}$$

$$= u - U(I + CS_KU)^{-1} CS_K u, \tag{25}$$

which is Equation 13.

A.4 Reference-preconditioner application and operation count

The cache in Equation 12 can be assembled without forming a dense activity matrix. Define output-to-layer transports

$$P_K = C, \qquad P_\ell = P_{\ell+1} J_\ell, \tag{26}$$

and compute the blocks of U by

$$U_1 = P_1^\top, \qquad U_{\ell+1} = P_{\ell+1}^\top + J_\ell U_\ell, \qquad M = I + CU_K. \tag{27}$$

Only the J_ℓ, U_ℓ, and a Cholesky factor of M are retained.

For dense layers, minibatch size B, hidden width N, and output size O, the setup and direction costs under our source-operation convention are

$$S_{\mathrm{CBT}} = 4(K-1)ON^2 + 2O^2N + \tfrac{1}{3}O^3, \tag{28}$$

$$D_{\mathrm{CBT}} = 4B(K-1)N^2 + 2BON + 2BO^2 + 2BKON. \tag{29}$$

The ledger counts a dense multiply–add as two FLOPs. Standard dense Cholesky uses $O^3/3$ FLOPs at leading order (equivalently, $O^3/6$ multiply–add pairs); no other cubic output-space operation is executed here. Lower-order terms are omitted consistently. This is the same convention used for every reported method and baseline. Both are linear in K at fixed B, N, O. Setup propagates O output-space right-hand sides, while each solve propagates B gradient right-hand sides. By contrast, T activity-GD steps traverse the chain T times; choosing $T = K$ makes that activity work quadratic in depth at fixed width.

A.5 Exact-Hessian width cost

At $K = 128$, batch size 64, and ten outputs, the source-operation ledger gives the following cost ratios for one per-sample exact-Hessian Newton iteration over one complete CBT update. This is a cost-only calculation under the same dense-operation convention; it does not assign an accuracy endpoint to the widths. Both assumptions favor Newton: CBT is charged for its worst observed path, with the secondary candidate and the restricted fallback both executed, whereas Newton is charged for a single current-curvature iteration, although it needs many iterations to converge.

Hidden width N	16	32	64	128	256
Newton / CBT FLOPs	2.03	3.96	7.91	15.85	31.77

For nonlinear least squares with residual vector $\mathbf{r}(z)$ and residual Jacobian $\mathcal{J}(z)$,

$$\nabla^2 \mathcal{F}(z) = \mathcal{J}(z)^\top \mathcal{J}(z) + \sum_i \mathbf{r}_i(z) \nabla^2 \mathbf{r}_i(z). \tag{30}$$

Equation 13 applies the inverse of the batch-shared reference preconditioner up to floating-point error. This preconditioner omits sample-to-sample Jacobian variation, changes in the Jacobian away from z_{ff}, and the residual-weighted second-derivative term in Equation 30. It is therefore the reference preconditioner for the nonlinear problem, not its Hessian.

A.6 Restricted Gauss–Newton fallback

At a candidate p satisfying the energy acceptance criterion, write the nonlinear residual vector as $\mathbf{r}(z)$. Its Gauss–Newton quadratic model is

$$Q_p(\delta) = \mathcal{F}(p) + g(p)^\top \delta + \frac{1}{2} \delta^\top G(p) \delta, \qquad G(p) = \mathcal{J}(p)^\top \mathcal{J}(p). \tag{31}$$

Let E_{S_h} inject a vector supported on the last h activity blocks. The minimizer of the restricted quadratic model is

$$\delta_{S_h} = -E_{S_h} \left(E_{S_h}^\top G(p) E_{S_h} \right)^{-1} E_{S_h}^\top g(p). \tag{32}$$

The implementation obtains this vector with the same two recurrences using sample-specific current Jacobians. Moreover, $g(p)^\top \delta_{S_h} = -(E_{S_h}^\top g(p))^\top (E_{S_h}^\top G(p) E_{S_h})^{-1} (E_{S_h}^\top g(p)) \leq 0$ because the hidden residual chain makes $G(p)$ positive definite. The inequality is strict exactly when $E_{S_h}^\top g(p) \neq 0$; if this projected gradient is zero, then $\delta_{S_h} = 0$. This directional result implies decrease for sufficiently small positive scales, not safety of every scale in the finite trial set. The support S_h, trial scales, and the energy acceptance criterion remain algorithmic policies.

B Experimental Protocol

B.1 Fixed solver constants

Table 1 lists the constants shared by every reported CBT run.

Table 1: Fixed solver and training constants used in the reported experiments.

Component	Fixed value
Primary step size α_0	0.2
Secondary step size α_1	0.02
GD refinement	3 steps, step size 0.3
Fallback region	last 16 activity blocks
Fallback Jacobian	current, per sample
Armijo trial scales	$1, 1/2, 1/4, 1/8, 1/16$
Armijo constant	10^{-4}
Energy tolerance	$10^{-10} \max(1, \lvert\mathcal{F}(z_{\text{ff}})\rvert)$
Outer optimizer	Adam, learning rate 10^{-3}

The fallback is triggered exactly when the primary candidate fails the energy acceptance criterion for at least one sample. The trigger does not use the task, activation, epoch, seed, validation result, history, a learned threshold, or a parameter-gradient diagnostic. When the branch is skipped, its FLOPs are not charged.

B.2 Training, selection, and accounting

All training comparisons use batch size 64, Adam with learning rate 10^{-3}, float64 arithmetic, and seeds 0, 1, and 2. Paired runs reuse the same initialization, deterministic train/validation split, and epochwise minibatch permutations. A trajectory may stop after epoch 5 when validation accuracy has failed to improve by 0.1 points for three checks and the last three epoch-end accuracies span at

most 0.5 points; otherwise it is capped at 60 epochs. We restore the best complete validation epoch and evaluate the test set exactly once.

All GD and HB activity updates use step size 0.3, fixed across depths and selected before the reported experiments; HB initializes its momentum buffer at zero and uses momentum 0.9. GD-CM and HB-CM use integer iteration budgets selected on a disjoint calibration seed to match the worst-case analytical FLOP estimate of CBT, without examining validation or test accuracy. Table 2 lists the resulting number of activity steps per training update in the depth and width experiments; HB-CM uses the same budget as GD-CM with momentum 0.9. GD-K instead takes one activity step per layer. Training baselines execute their fixed budgets without early stopping or step rejection; only the shared outer training plateau rule stops a trajectory. The FLOP accounting model counts all executed solver branches, cache construction, parameter gradients, warm-up, validation, and the final test. GPU time is reported only for runs on the same NVIDIA A100.

Table 2: Activity steps per training update for the first-order baselines in the compute-matched MNIST depth and width experiments. GD-CM and HB-CM budgets T are fixed from the worst-case CBT update before training; a T-step update costs $T + 3.5$ GE, where the constant is the per-update work outside the activity steps. The CBT column gives the realised per-update cost, averaged over training and given as the range over three seeds. GD-K takes one step per hidden layer (GD-128 in the width study). The width study has no HB arm.

Setting	K	N	GD-CM/HB-CM T	GE/update at T	CBT GE/update	GD-K steps
MNIST, depth	32	64	13	16.5	7.41–9.38	32
	64	64	11	14.5	9.93–13.14	64
	128	64	10	13.5	12.84–12.96	128
	256	64	9	12.5	12.45–12.50	256
MNIST, width	128	128	10	13.5	9.23–11.73	128
	128	256	10	13.5	8.69–9.72	128

The worst-case calibration is conservative for the baselines. The realised baseline/CBT per-update ratio is 1.76–2.23 at $K = 32$, 1.10–1.46 at $K = 64$, 1.04–1.05 at $K = 128$, and 1.000–1.004 at $K = 256$; it is 1.15–1.55 in the width study.

The epoch-zero mechanism controls use the same GD step 0.3 and HB momentum 0.9, start from identical feedforward activities, and stop at the first batch-mean energy no greater than the corresponding final CBT energy (right-censored after 20,000 iterations). The second-order comparator is *regularized exact-Hessian Newton*. For each sample and attempt, the implementation forms $H_b(z) = \nabla_z^2 \mathcal{F}_b(z)$ with the residual-weighted second-derivative term in Equation 30, then solves $(H_b(z) + \lambda_b I)d_b = g_b$ by block Cholesky and proposes $z_b - d_b$. It therefore does not use the Gauss–Newton/LM matrix $\mathcal{J}_b^\top \mathcal{J}_b + \lambda_b I$. The diagonal regularization parameter starts at 10^{-3}, is multiplied by 10 after a non-finite or non-decreasing trial, divided by 10 after acceptance, and clipped to $[10^{-10}, 10^{10}]$. Rejected trials retain the current activity but count their gradient, Hessian construction, factorization, solve, and energy evaluation. The merged depth study permits at most 512 uninterrupted attempts and stops when the relative activity-gradient norm falls below 10^{-7}; the reused $K = 256$ runs used the stricter 10^{-8} criterion and at most 256 attempts. The reported cost to the CBT target is the first crossing of the per-seed final CBT energy; it is not the cost of running the comparator to its own relative-gradient stopping criterion.

MNIST and Fashion-MNIST use their established 60,000/10,000 train/test splits; the training partition is deterministically split into training and validation subsets. Fashion-MNIST has the same image shape and split sizes as MNIST (Xiao et al., 2017). CIFAR-10 has 50,000 training and 10,000 test images (Krizhevsky et al., 2009); the residual-MLP transfer flattens each $32 \times 32 \times 3$ image and applies per-channel normalization without augmentation. This fully connected model is used only to test whether the solver comparison persists after changing the dataset and input dimension.

C Trained-Checkpoint Learning-Signal Audit

We freeze the selected $K = 128, N = 64$ tanh checkpoint from each training seed and use one deterministic 16-example validation batch per seed, without a parameter update. A per-sample

regularized exact-Hessian Newton solve is continued to relative activity-gradient residual below 5.0×10^{-10} and defines the oracle endpoint. GD and HB are truncated at the largest whole-step prefix not exceeding the complete-update FLOPs of CBT. Oracle energy quality is the fraction of the feedforward-to-oracle decrease recovered by each arm.

Table 3: Per-seed trained-checkpoint endpoints. q_E is oracle energy quality; $\cos_W$ is cosine of the parameter gradient with the oracle gradient.

	q_E				$\cos_W$			
Seed	FF	GD	HB	CBT	FF	GD	HB	CBT
0	0	.065	.156	.567	.330	.340	.357	.594
1	0	.066	.154	.508	.324	.330	.339	.613
2	0	.038	.074	.385	.361	.364	.367	.541

Table 3 lists the per-seed endpoints. The mean oracle energy qualities are $48.7\% \pm 9.3\%$ for CBT, $5.6\% \pm 1.6\%$ for GD, and $12.8\% \pm 4.7\%$ for HB. The first Newton prefix whose energy quality reaches the CBT endpoint costs 30.09, 30.09, and 25.79 times the CBT complete-update FLOPs for seeds 0, 1, and 2, respectively. Energy quality and gradient fidelity are distinct: for example, the quality-matched Newton prefix reaches roughly 0.61 oracle energy quality in the first two seeds while its gradient cosine remains 0.57 and 0.52. Averaged over seeds, this quality-matched prefix has gradient cosine 0.54, compared with 0.58 for CBT.

D STOPPING-RULE-INDEPENDENT LEARNING EFFICIENCY

Stopping-rule-independent efficiency. Figure 5 shows that the efficiency conclusion is visible throughout training rather than being created by the plateau rule. At the stable 85% target, GD-CM requires $23.3\times$, $38.2\times$, $46.6\times$, and $43.8\times$ the gradient-equivalents of CBT at $K = 32, 64, 128, 256$, respectively. The corresponding baseline-relative time ratios are $7.3\times$, $11.1\times$, $18.7\times$, and $24.2\times$. HB-CM is right-censored at the two deepest settings in Figure 6; hollow markers show these incomplete groups rather than averaging only the successful seeds.

Systems costs. CBT has a larger one-time compilation/warm-up cost, while its steady-state complete-update latency remains below GD-CM at every measured depth (Figure 7).

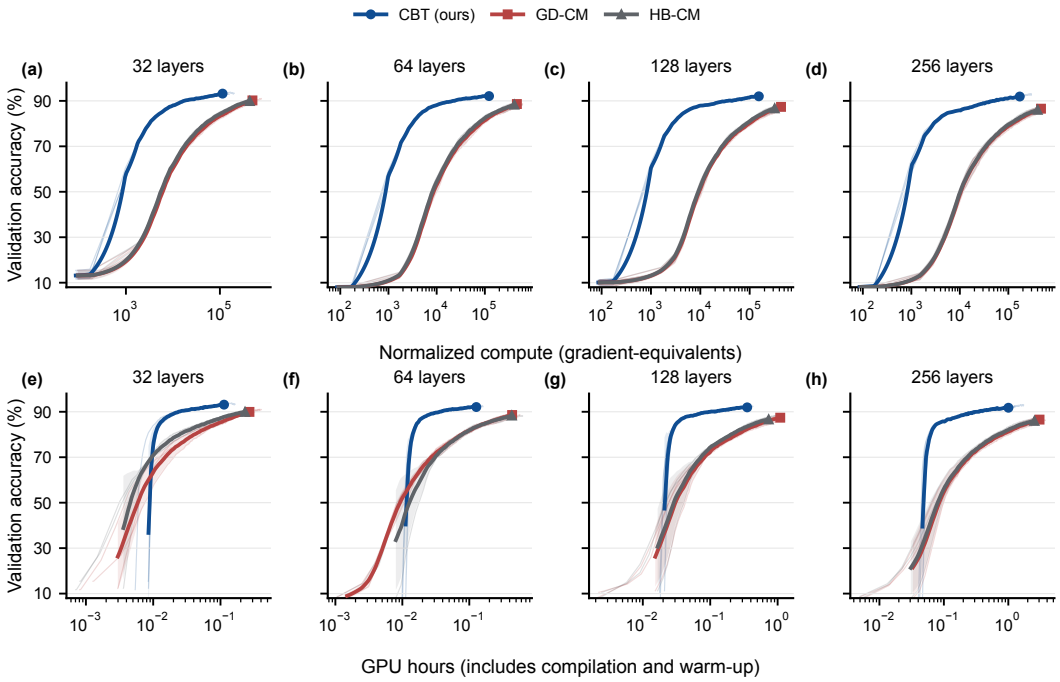


Figure 5: **Training trajectories do not depend on the plateau stopping rule.** Validation accuracy is plotted against normalized compute in cumulative gradient-equivalents (top) and baseline-relative GPU hours including compilation and warm-up (bottom). Thin lines are individual seeds; thick lines and bands are the mean and one sample standard deviation after interpolation over the resource axis. The balanced comparison uses three seeds for CBT, compute-matched GD (GD-CM), and compute-matched heavy ball (HB-CM) at every depth.

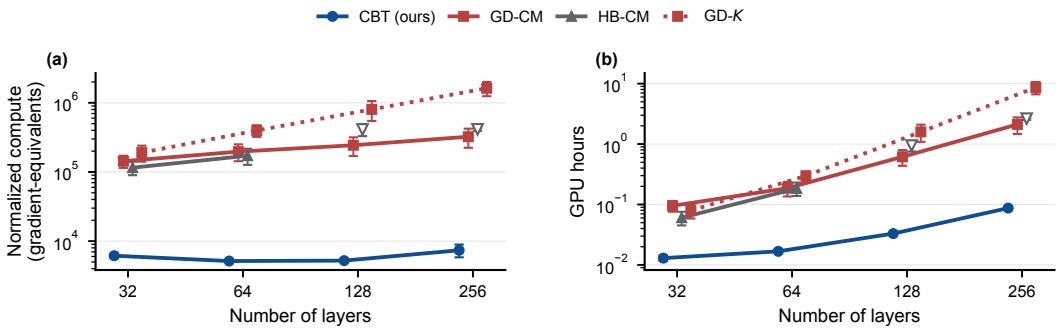


Figure 6: **Time to stable 85% validation accuracy.** Stable attainment is the first validation evaluation at or above 85% whose next evaluation also remains above 85%. Filled markers show mean ± one sample standard deviation when all three seeds reach the target. A hollow downward triangle denotes a right-censored group in which fewer than three seeds reach the target and is placed at its mean stopping budget. GPU hours include compilation and warm-up and are baseline-relative.

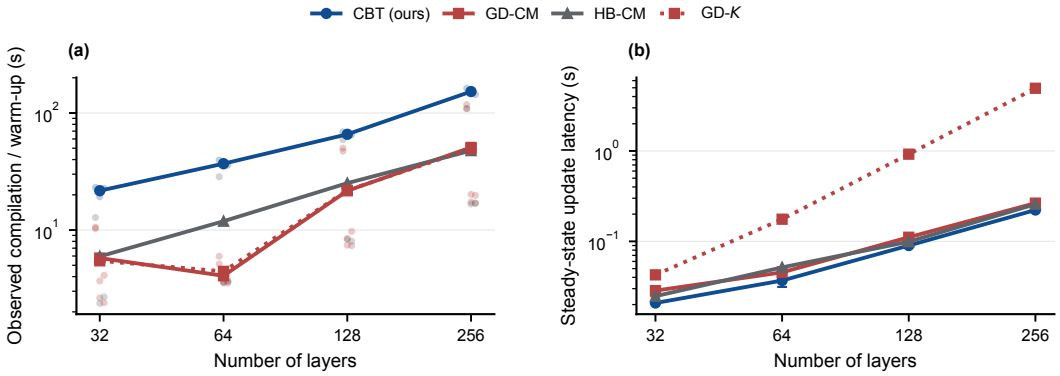


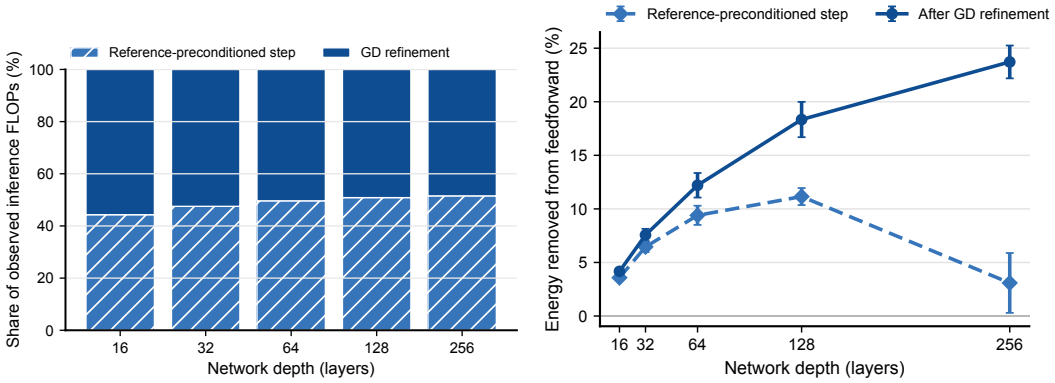
Figure 7: **Compilation and steady-state update latency.** The left panel reports the observed compilation/warm-up cost for each run, with translucent points for individual seeds and a line through their arithmetic mean. These values may include persistent-cache and execution-order effects and are therefore descriptive rather than a hardware-absolute benchmark. The right panel reports mean ± one sample standard deviation of the steady-state complete-update latency.

E Epoch-Zero Mechanism Audit

The mechanism study fixes one deterministic eight-example MNIST batch and probes five independently initialized, untrained tanh residual μPC MLPs at each depth. The regularized exact-Hessian Newton reference is computed per sample; all five $K = 256$ runs terminate with relative gradient residual between 3.10×10^{-13} and 3.63×10^{-9}. Comparator FLOPs include every Hessian construction, factorization, solve, rejected attempt, and evaluation of the energy acceptance criterion.

The stage audit is a descriptive post-hoc analysis of the stored solver traces. It separates the reference-preconditioned step from the subsequent GD refinement. At $K = 256$, the chain step removes $3.09\% \pm 2.80\%$ of feedforward energy and the three refinements raise the total to $23.72\% \pm 1.53\%$. Applied directly at feedforward, three GD steps remove $0.730\% \pm 0.076\%$; at approximately the same 7-GE cost as the complete iterate, GD removes $1.24\% \pm 0.13\%$ and HB $2.40\% \pm 0.25\%$. Across depths, the three post-solve refinements supply the larger raw decrease (Figure 8b), while the reference step accounts for 44–51% of observed inference FLOPs through the gradient, cache setup, direction application, and initial energy evaluation (Figure 8a). The fallback executes in none of the 25 runs; its role is evaluated at trained checkpoints in Section 4.4.

Normalization by one depth-matched activity gradient removes the unavoidable linear cost of visiting more layers. Raw inference FLOPs still increase with depth for CBT, GD, and HB (Figure 9); the bounded quantity is cost per depth-matched gradient evaluation. The observed final iterate remains below the deterministic all-branch bound, which decreases from 21.260 GE at $K = 16$ to 14.590 GE at $K = 256$. The regularized exact-Hessian Newton curve in Figure 2b records first passage to the common CBT-energy target. Running Newton to its relative-gradient stopping criterion reaches lower energy but costs 1048.3 GE at $K = 16$ and 7528.9 ± 1530.3 GE at $K = 256$.



(a) Allocation of observed inference FLOPs.

(b) Cumulative energy removed from feedforward.

Figure 8: **Stage attribution for the epoch-zero mechanism study.** Means and sample standard deviations use five network seeds. Only distinct observed iterates are shown; dormant fallback stages are not plotted as overlapping progress.

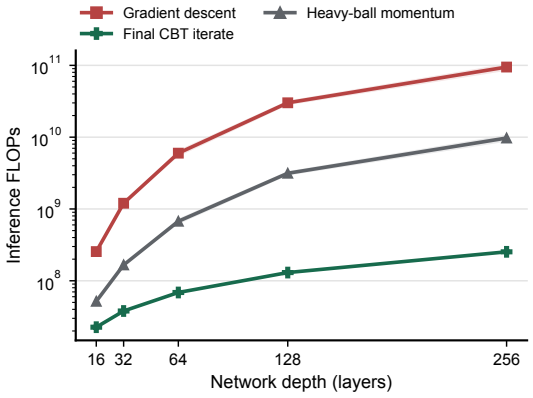


Figure 9: **Raw inference work in the epoch-zero depth probe.** CBT, GD, and HB are evaluated at the same per-seed final CBT-energy target. Mean $\pm$ s.d. over five network seeds. Unlike the GE-normalized result, the raw cost of every method increases with depth.

F Width-Scaling Frontier

G Nonlinear Safeguard Ablations

Over the full validation set of the $K = 64$ tanh checkpoints, the energy trigger skips the fallback on 54 of 234 batches, and conditional execution uses 94.0% of the always-on fallback FLOPs while returning activities that satisfy the energy acceptance criterion in every batch. The fallback is therefore best understood as a robustness mechanism.

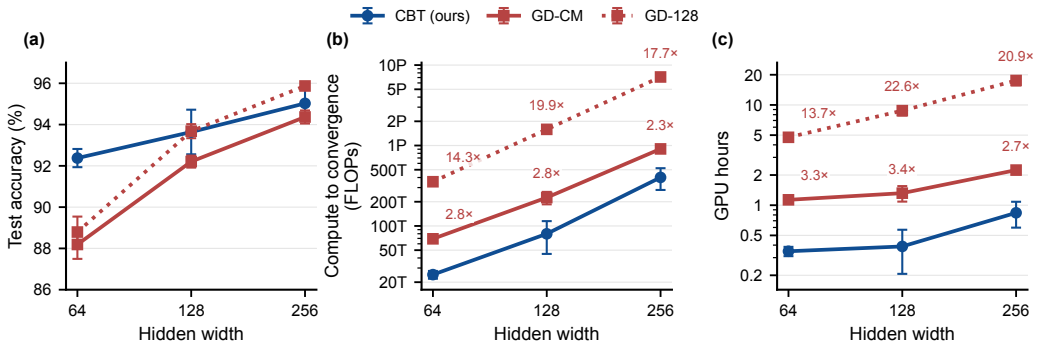


Figure 10: **Width scaling at 128 layers.** Test accuracy, total dominant FLOPs, and baseline-relative complete-update A100 time are mean ± s.d. over three seeds. Labels report each baseline/CBT cost ratio.

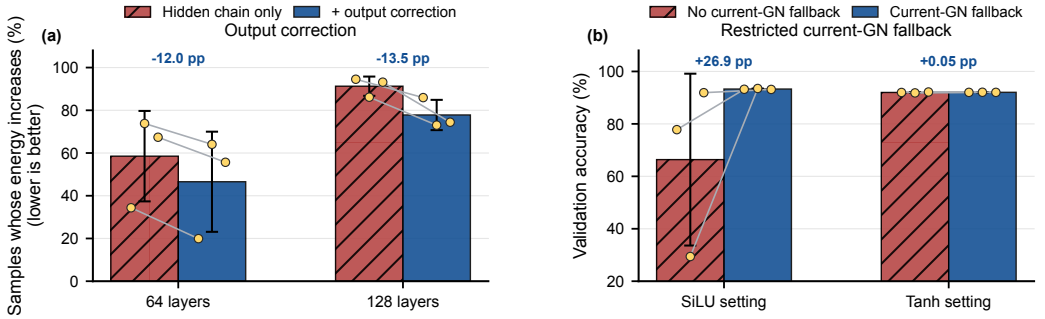


Figure 11: **The two components address different errors.** Left: at fixed trained states, adding the output correction reduces the fraction of pre-refinement proposals that increase energy. Right: the restricted Gauss–Newton fallback substantially improves validation accuracy in the tested SiLU setting and has little effect in the tested tanh setting. Bars show means, error bars show one s.d., dots are seeds, and lines retain pairing.

H Transfer within the Residual Family

The solver configuration is unchanged across the following studies. We separate dataset and activation transfer because they test different sources of variation and have different baseline coverage.

H.1 Dataset transfer

Table 4: Dataset transfer with residual MLPs. Values are validation-selected test accuracy, mean ± s.d. over three seeds. No solver parameter is retuned.

Dataset	K	N	CBT	GD-CM	GD-K
Fashion-MNIST	64	64	0.8297 ± 0.0043	0.7959 ± 0.0055	0.7932 ± 0.0017
CIFAR-10	64	128	0.4075 ± 0.0038	0.3757 ± 0.0096	0.3779 ± 0.0082

As Table 4 shows, the fixed configuration improves over GD-CM by 3.38 points on Fashion-MNIST and 3.18 points on CIFAR-10; all six paired CIFAR-10 seed differences are positive. On CIFAR-10, GD-CM and GD-K require $2.68\times$ and $7.43\times$ the total GE of CBT, respectively. The smaller matched-GD ratio than on canonical MNIST is expected because the wider input layer moves more work outside the repeated hidden chain. This is a solver-transfer test: the architecture remains a residual MLP while the dataset and input dimension change.

H.2 ACTIVATION TRANSFER

Table 5: Activation transfer for the unchanged CBT configuration on MNIST, $K = 128, N = 64$. Both rows satisfy the common plateau rule and the energy acceptance criterion.

Activation	Test accuracy	Epochs to plateau
tanh	0.9238 ± 0.0044	14.3 ± 1.5
SiLU	0.9337 ± 0.0107	13.7 ± 5.9

All six trajectories plateau before the 60-epoch cap, and both the executed and skipped fallback branches occur for every activation and seed. The result shows that the fixed configuration satisfies the energy acceptance criterion with tanh and SiLU, including the activation for which the fallback is most consequential in Figure 11. Table 5 thus shows that one fixed configuration adapts across activations.

I ARCHITECTURE SCOPE: MODELS WITHOUT RESIDUAL CONNECTIONS

Removing every identity skip tests the mechanism directly, because the output correction propagates through products of layer Jacobians. Table 6 reports the learning control; all final activities remain finite and satisfy the energy acceptance criterion, but 7 of 18 runs reach the 60-epoch cap.

Table 6: MNIST/tanh test accuracy without residual connections, mean $\pm$ s.d. over three seeds. The unchanged CBT configuration attains lower accuracy than both first-order baselines.

K	CBT	GD-CM	GD-K
32	0.4798 ± 0.0940	0.5608 ± 0.0328	0.6485 ± 0.0390
64	0.2632 ± 0.0347	0.3974 ± 0.0407	0.4311 ± 0.0394

The accompanying fixed-weight probe explains the reversal. From $K = 8$ to 128, the end-to-end reference gain $\| \prod_\ell J_\ell \|_2$ remains 2.05–2.26 with residual skips but falls from 0.91 to 0.14 without them; descriptive fits give $K^{0.03}$ and $K^{-0.68}$, respectively. The training-time fallback trigger likewise rises from 0.138 ± 0.117 for the residual $K = 32$ model to 0.918 ± 0.071 without skips. The cached output correction therefore relies on a non-collapsing path through the layer Jacobians, which residual connections provide.