CIRCULAR PREDICTIVE CODING MODELS UNLOCK TEST-TIME COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Predictive coding and equilibrium propagation (EP) perform credit assignment through an energy minimization process. Despite their ability to train models with arbitrary network topologies, they have mostly been used to train feedforward architectures to match the performance of backprop. Here, we use these methods to enable iterative algorithmic reasoning through energy minimization. By closing a feedforward model into a loop, done by identifying its first and last hidden layers, effective recurrent computation emerges from the continuous relaxation of the energy landscape. We show that in predictive coding networks with this circular structure, the bi-level optimization framework of EP provably approximates the gradient update of an implicit layer, allowing it to train a model whose computational depth can increase at test time by relaxing for a longer time for more complex problems. The same construction allows us to achieve image classification performance comparable to standard predictive coding ResNet models while using substantially fewer parameters thanks to the effective depth, and to train a recursive transformer on the Tiny Stories dataset. This construction has direct implications for physical neural networks, where inference and learning are carried out by the dynamics of the substrate itself, and where computational depth could be traded for recursive operations on a fixed-size circuit.

1 INTRODUCTION

Predictive coding (PC) Rao & Ballard (1999); Salvatori et al. (2023a) and equilibrium propagation (EP) Scellier & Bengio (2017) are learning algorithms for neural networks that perform credit assignment via brain-inspired computation. Both methods rely on the relaxation of an energy functional defined over neurons, whose computation are local in space and time, and can be used to train neural architectures with any kind of topology, including looped ones Salvatori et al. (2022a). Yet, these properties have rarely been exploited computationally, as recent work has largely used this relaxation to reach the equilibrium needed for credit assignment of feedforward architectures with a fixed number of layers Scellier et al. (2024); Pinchetti et al. (2025). This choice was mostly pragmatic: when the objective is to reproduce results obtained via backpropagation, we have access to a large body of results on architectures, initialization, and optimization that can be used almost unchanged. On the one hand, this has been greatly beneficial, as in a couple of years, a small community managed to scale up predictive coding models from the MNIST dataset to ImageNeT Whittington & Bogacz (2017); Kerjan et al. (2026); on the other, this made researchers overlook interesting research directions, such as the study of novel architectures and different ways to exploit the energy minimization process of this class of algorithms. In this work, we address this by asking the following:

Question. Can energy-based relaxation be turned into a form of test-time computation, and what architectural structure is required for such behavior to emerge?

This question is particularly relevant for physical neural networks (PNNs), in which computation is performed directly by the dynamics of an electronic, optical, or mechanical system rather than simulated on digital hardware Wright et al. (2022); Momeni et al. (2025); Kendall et al. (2020); Yi et al. (2023): There, depth is expensive, as every additional layer is additional hardware, with corresponding costs in area, energy, and device noise. A positive answer would therefore let a fixed

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.

physical circuit trade replicated depth for relaxation time. This question could also be interesting for neuroscientists, as it has been shown that the brain solves harder problems by processing for longer rather than by recruiting more areas Gold & Shadlen (2007); Kar et al. (2019), and its anatomy is rich in closed loops, such as cortico-basal ganglia and cortico-cerebellar circuits Alexander et al. (1986); Kelly & Strick (2003). Yet, predictive coding models of cortex are typically hierarchical Friston (2005), and showing that the same dynamics on a loop support iterative computation would offer a mechanism by which depth can be defined via settling time rather than by anatomy.

Our answer is that it suffices to close a feedforward PC model into a loop, by merging its first and last hidden layers, and to attach an encoder and a decoder to the loop. To formally show that this is enough, we link this new kind of circular predictive coding (CPC) models to deep equilibrium models (DEQs) Bai et al. (2019), implicit networks whose computation is defined by convergence to a fixed point, and whose depth grows by iterating the underlying transformation for longer Anil et al. (2022). To further connect not only how such models perform inference, but also how they update their parameters, we show that training such a model using the contrastive signal of equilibrium propagation allows us to approximate the weight update of DEQs Kerjan et al. (2026); Zucchet & Sacramento (2022).

While in standard recursive models the main goals are reasoning and parameter efficiency, in predictive coding this may tackle another, long-standing problem of the field: the training of very deep neural networks, hard to train due to a kind of *vanishing gradient* problem of the energy propagation Pinchetti et al. (2025); Innocenti et al. (2026); Qi et al. (2025). Circular models sidestep this issue by construction: their effective depth is determined by the relaxation time, and can hence be arbitrary and task dependent, while the error signal only travels around a loop of at most three hidden layers, so that no parameter is ever more than a few vertices away from the nudged output. This is reflected in the better performance of our model in image classification tasks. Our contributions are briefly as follows:

- We prove and provide empirical evidence that (i) the free phase of a CPC model has a unique energy minimum that coincides with the fixed point of an equivalent DEQ, hence identifying the solution of the inference phase of the two models; and (ii) use the result of Zucchet & Sacramento (2022) to show that when performing an update of the parameters, EP approximates the implicit gradient of the corresponding deep equilibrium model;
- We evaluate the test-time computation properties of the circular model on two tasks: Sudoku and Prefix-sum. In Sudokus, complex puzzles require a larger amount of iterations to be solved; in Prefix-sum, the model is able to generalize to longer sequences than the ones it was trained on. This is the first time predictive coding, or other learning algorithms with local computations, have been applied to algorithmic and constrained reasoning tasks;
- To study the performance of this model in more standard machine learning tasks, we also train it on CIFAR10/CIFAR100, showing comparable performance with much larger models such as PC-based ResNet18, and on autoregressive language modelling on the Tiny stories dataset (Eldan & Li, 2023), the largest NLP task performed on PC models, almost matching the performance of a deep equilibrium model of the same size.

2 Related Works

The formulation of predictive coding used in machine learning originates from models of visual cortex (Rao & Ballard, 1999), and was first turned into a supervised learning algorithm by showing that its weight updates approximate those of backprop in multilayer perceptrons (Whittington & Bogacz, 2017), a result later extended to arbitrary computational graphs (Millidge et al., 2022). These results, however, still require a directed acyclic structure, and hence inherit the main limitation of backprop: models must be feedforward. This restriction can be removed by defining the energy over a generic graph, where every neuron is predicted by the activity of the whole network, so that the same local learning rules can train structures with cycles and backward connections (Salvatori et al., 2022a). This property has since been used to combine discriminative and generative processing (Oliviers et al., 2026), to perform causal inference (Salvatori et al., 2023b), and to study how feedback loops create internal contradictions that can stall learning (Seely, 2025). Yet, besides very small scale tasks and a few theoretical results, no real use has been found for it. Our work provides one: closing a

feedforward model into a circle turns the relaxation of the energy into a form of iterative computation, equivalent to that of deep equilibrium models (Bai et al., 2019).

A second line of research has focused on depth. Predictive coding models are hard to train as the number of layers grows, as prediction errors concentrate near the output and leave early layers with almost no learning signal (Pinchetti et al., 2025). On the one hand, this problem has been addressed formally, via depth-stable parametrizations (Innocenti et al., 2026) or layer-local multipliers that recover exact backprop gradients (Seely & Gould, 2026), but only on MNIST-scale tasks, with no evidence of performance at scale. On the other, *ad-hoc* solutions perform very well on CIFAR10/100 (Ha et al., 2026; Qi et al., 2025), but mitigate the symptoms rather than solving the underlying problem. Our circular models sidestep this issue by construction: their effective depth is determined by the relaxation time, while the error signal only travels around a loop of at most three hidden layers.

3 BACKGROUND

In this section, we review the concept of *predictive coding graphs*, a generalization of predictive coding networks to models with arbitrary topologies, where vertices correspond to a group of neurons Salvatori et al. (2022a). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph, whose vertices we index by $i \in \mathcal{V}$, and whose edges $(j, i) \in \mathcal{E}$ carry a parametric map $g_{j,i}$. Coherently with the predictive coding literature, every vertex encodes three quantities: a random variable called value node $\mathbf{x}_i \in \mathbb{R}^{d_i}$, which represents the current neural activity of the vertex; a *prediction*

$$\hat{\mathbf{x}}_i = \sum_{j \in \mathrm{pa}(i)} g_{j,i}(\mathbf{x}_j), \tag{1}$$

computed from the value nodes of its parents along the incoming edges; and an error node $\boldsymbol{\varepsilon}_i = \mathbf{x}_i - \hat{\mathbf{x}}_i$, defined as the mismatch between the activity of the vertex and the signals it receives. Collecting all value nodes into a single vector $\mathbf{x} = (\mathbf{x}_i)_{i \in \mathcal{V}}$, and using θ to represent the joint parameters of the parametric maps, the system is governed by an energy given by the sum of the squared prediction errors, which results in the following Lyapunov function for the dynamics:

$$E(\theta, \mathbf{x}) = \frac{1}{2} \sum_{\mathcal{V}} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 = \frac{1}{2} \sum_{\mathcal{V}} \|\boldsymbol{\varepsilon}_i\|_2^2. \tag{2}$$

Inference. As PC graphs can be seen as a physical system governed by the above energy function, it is possible to present an input signal to the model by imposing boundary conditions on one vertex of the model. This is analogous to fixing the potentials of some nodes of a resistive network with voltage sources. The remaining value nodes are then updated following the gradient flow of the energy until they reach a free equilibrium $\mathbf{x}^*$, that is, a state at which $\nabla_{\mathbf{x}_{\text{free}}} E(\theta, \mathbf{x}^*) = 0$. The prediction of the model is then read from a subset of vertices $\mathcal{O} \subset \mathcal{V}$, the *output vertices*, as their value at convergence, $\mathbf{x}_{\mathcal{O}}$. This mirrors measuring the potential at other nodes of the circuit, or the displacement of other nodes of the elastic network, once the system has settled.

Learning. While the inference rule intuitively follows from the idea of a dynamical system converging to an equilibrium, understanding how the parameters of such a model can be trained is less straightforward. There are two main techniques in the literature: the standard update of predictive coding, and that of *equilibrium propagation* (EP), a more general method defined to train arbitrary physical systems, of which a specific formulation for predictive coding has been recently derived Kerjan et al. (2026). In this work, we will use the second one.

Once a free equilibrium has been reached, EP can be used to estimate the gradient of the loss by contrasting it against a second weakly perturbed equilibrium, rather than by differentiating through the inference dynamics that produced it. Let $C(\mathbf{x}_{\mathcal{O}}, \mathbf{y})$ be a loss function that compares the value nodes of the output vertices to the target $\mathbf{y}$, and let $\beta \in \mathbb{R}$. Let us assume we have already performed the inference phase over a given input $\mathbf{x}_{\mathcal{O}}$, and that we have converged to an equilibrium state $\mathbf{x}^*$ The loss is then applied as a weak external force acting on the output vertices, giving the energy $F_\beta(\theta, \mathbf{x}, \mathbf{y}) = E(\theta, \mathbf{x}) + \beta\, C(\mathbf{x}_{\mathcal{O}}, \mathbf{y})$. Keeping the input vertices clamped to $\mathbf{o}$ and letting the free value nodes relax on F_β, starting from $\mathbf{x}^*$, yields a second equilibrium point $\mathbf{x}^\beta$. The parameters are then updated via contrastive learning as follows:

$$G_{\mathrm{EP}}(\beta) = \frac{1}{\beta} \left[\nabla_\theta E(\theta, \mathbf{x}^\beta) - \nabla_\theta E(\theta, \mathbf{x}^*) \right]. \tag{3}$$

Nudging schemes. The above equation is a finite-difference approximation of the derivative of the loss function. Different choices of the sign of β therefore result in different estimators of the same gradient. The simplest one, *positive nudging*, uses a single $\beta > 0$, which pulls the output vertices towards the target. This is a forward difference, and its bias with respect to the true gradient is of order $O(\beta)$. *Centered nudging* Laborieux et al. (2021) removes this first-order term by relaxing to two equilibria, $\mathbf{x}^{\beta}$ and $\mathbf{x}^{-\beta}$, obtained by nudging the output vertices towards and away from the target, and contrasting them as follows:

$$G^{\mathrm{c}}_{\mathrm{EP}}(\beta) = \frac{1}{2\beta}\left[\nabla_\theta E(\theta, \mathbf{x}^{\beta}) - \nabla_\theta E(\theta, \mathbf{x}^{-\beta})\right], \tag{4}$$

whose bias is of order $O(\beta^2)$, at the cost of an additional nudged phase. *Random nudging* Scellier & Bengio (2017) keeps a single nudged phase, but samples the sign of β uniformly at random for every data point. In PC networks, these schemes have a particularly convenient form: as in the free equilibrium the value nodes coincide with their predictions, the free state has zero energy, hence reducing the update to $\nabla_\theta E(\theta, \mathbf{x}^*) = 0$ Kerjan et al. (2026).

3.1 Deep equilibrium models

A deep equilibrium (DEQ) model Bai et al. (2019) replaces a stack of layers by a single map $T_{\theta,\mathbf{o}} : \mathbb{R}^n \to \mathbb{R}^n$, applied to its own output: given an observation $\mathbf{o}$, the representation of the network is defined implicitly as a fixed point $\mathbf{z}^\star = T_{\theta,\mathbf{o}}(\mathbf{z}^\star)$, from which the prediction is read by a linear classifier. Iterating $T_{\theta,\mathbf{o}}$ from an arbitrary initial state converges to the fixed point $\mathbf{z}^\star$ if and only if the map is contractive. This process is conceptually equivalent to the forward pass through an infinitely deep network with tied weights, since every iteration applies the same transformation. Depth therefore becomes a runtime quantity rather than an architectural constant, and test-time compute is unlocked by providing more iterations than those used during training Anil et al. (2022).

Learning. As the length of the model is not fixed, training cannot be performed by unrolling the computational graph and using backprop through time. Furthermore, even if we had decided a priori for a fixed number of steps T, this would have $\mathcal{O}(T)$ memory complexity. Instead, one differentiates the fixed-point condition itself, which expresses the gradient of a cost C at equilibrium as

$$\nabla_\theta C = (\partial_\theta T)^\top (\mathbf{I} - \mathbf{J}_T^\top)^{-1} \nabla_{\mathbf{z}} C, \tag{5}$$

with $\mathbf{J}_T$ the Jacobian of the map at $\mathbf{z}^\star$. The factor $(\mathbf{I} - \mathbf{J}_T^\top)^{-1}$ is a global linear solve coupling every unit of the network to every other, with $\mathcal{O}(1)$ complexity: it is what makes the implicit gradient exact at any depth, and also what a system with access only to its own local dynamics cannot compute. This is done via the implicit function theorem, already linked to EP in a previous work Zucchet & Sacramento (2022).

4 Circular Predictive Coding

Consider a PC graph with the hierarchical structure typical of multilayer neural networks: an ordered chain of vertices $1, \dots, L$ in which the prediction of vertex i reads only the value node of vertex $i-1$. On example of such a model can be seen in Figure 1(b). We define *circular predictive coding* (CPC) the model obtained by closing this chain into a loop: the prediction of vertex 1 reads the value node of vertex L. The graph is completed by two vertices outside the loop, an input vertex clamped to the observation $\mathbf{o}$, provided to the model via an embedding layer e, and an output vertex from which the cost is read, connected to the loop via a decoder D into vertex L and a decoder edge d_θ out of it. This model, depicted in Figure 1(c), has the following local predictions:

$$\hat{\mathbf{x}}_1 = g_{L,1}(\mathbf{x}_L), \qquad \hat{\mathbf{x}}_i = g_{i-1,i}(\mathbf{x}_{i-1}), \quad 2 \le i < L, \qquad \hat{\mathbf{x}}_L = g_{L-1,L}\big(\mathbf{x}_{L-1}, e_\theta(\mathbf{o})\big), \tag{6}$$

together with $\hat{\mathbf{x}}_{\mathrm{out}} = d(\mathbf{x}_L)$. Note that when describing the dynamics of the circular model, we will use $\mathbf{x_0}$ and $\mathbf{x_L}$ to define the same value node, that is the result of the identification of the actual nodes $\mathbf{x_0}$ and $\mathbf{x_L}$ of the feedforward model. Differently from standard PC graphs, here the observation is presented to the network as an additive constant to a specific value node. Hence, vertex L plays two roles at once: it receives the stimulus, and it closes the recurrent constraint. Its own inference dynamics make this explicit,

$$\dot{\mathbf{x}}_L = -\boldsymbol{\varepsilon}_L + \mathbf{J}_{g_{L,1}}(\mathbf{x}_L)^\top \boldsymbol{\varepsilon}_1 + \mathbf{J}_{d_\theta}(\mathbf{x}_L)^\top \boldsymbol{\varepsilon}_{\mathrm{out}}. \tag{7}$$

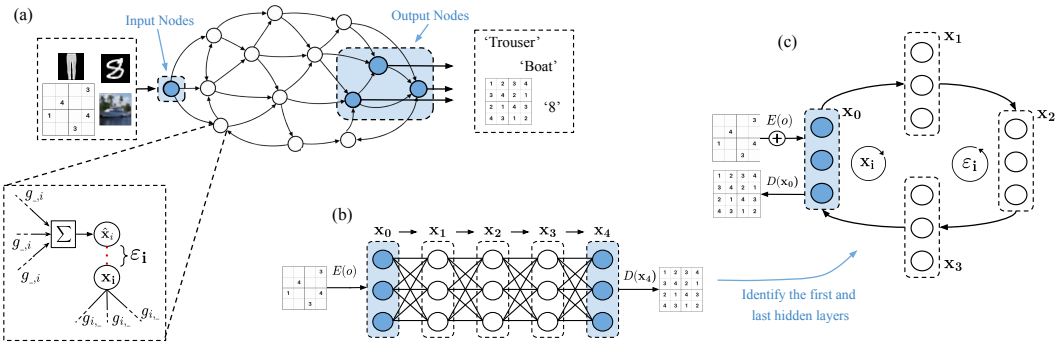


Figure 1: (a) An example of a PC graph. Zoomed in, the dynamics of every vertex, divided in value nodes, predictions, and error nodes; (b) An example of a feedforward predictive coding model; (c) The same feedforward model, where we have identified the first and last value nodes to create a circular structure, that will allow the model to learn and solve reasoning tasks.

4.1 Results

Two known facts that make PC models easy to train have to do with their free phase and weight update: First, the relaxation process of the free phase converges to a state of zero error, that is equivalent to the one reached by a standard forward pass. It is for this reason that most works avoid computing the free phase, and just perform a single pass. Second, their weight update is able to approximate that of backprop Salvatori et al. (2022b); Millidge et al. (2020). We now show that these properties that connect feedforward PC models and feedforward neural networks have an equivalent in circular PC models and DEQs. Every theorem in this section is presented informally for clarity of exposition and lack of space due to the page limit; we refer to the supplementary material for the formal statements and proofs.

Let us assume we have a circular PC model M with edge maps $g_{L,1}, g_{1,2}, \dots, g_{L-1,L}$ and encoder e_θ, as in Eq. 6, and let M' be the DEQ built from those same maps, whose implicit layer is their composition around the loop, $T_{\theta,\mathbf{o}}$. The two models are structurally identical: same layers and same parameters. They differ only in how they perform inference: M relaxes all of its value nodes along the gradient flow of the energy, while M' iterates $T_{\theta,\mathbf{o}}$ from the same starting point. Then,

Theorem 1 (Inference) *Assume that every map of the loop is C^1 with Lip. constant $q < 1$. Then:*

(1) The energy of M has a unique stationary point, which is its global minimum and has zero energy;

(2) The main value node of that state is the unique fixed point of $T_{\theta,\mathbf{o}}$.

The free phase of a circular model and the forward pass of $\mathrm{DEQ}(\theta)$ therefore return the same output, and the same latent variables: the relaxation reaches, node by node, the state that iterating the composed map reaches, and it cannot stall anywhere else. This is the counterpart, for circular models, of the fact that the free phase of a feedforward PC model converges to a state of zero energy, and produces the same output and hidden states of a forward pass. To then show a connection between how parameters are updated in the two methods, we need to rely on previous results, which have studied the relations between BP through time, EP, and the implicit function theorem Zucchet & Sacramento (2022); Scellier & Bengio (2017) [1].

Theorem 2 (Learning) *Let $\mathbf{x}^0$ be a free equilibrium at which the Hessian $\mathbf{H}$ of E in the free value nodes is positive definite, let E be twice continuously differentiable jointly in the parameters and the state near $\mathbf{x}^0$, and let C be twice continuously differentiable there. Write $\mathcal{L}(\theta) = C(\mathbf{x}^0_{\text{out}}(\theta), \mathbf{y})$ for the cost at that equilibrium. If both phases are solved exactly, then $G_{\text{EP}}(\beta) \to \nabla_\theta \mathcal{L}(\theta)$ as $\beta \to 0$.*

[1] It is because we need to approximate BP through time, and not standard BP, that we are using the weight update of EP instead of the usual one for PC models. Theorem 2 would have not held had we used the standard weight update of PC.

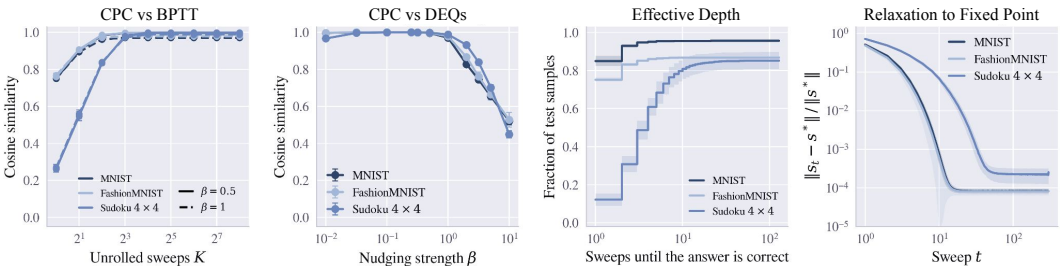


Figure 2: (a) Cosine similarity between the EP weight gradient and the gradient of a K-sweep unroll, on 128 samples. (b) Same EP gradients against the exact implicit-function gradient of a DEQ. Fidelity decays for large β, where the nudge moves the equilibrium beyond the linear regime EP relies on. (c) The sweep after which the loop's prediction is correct and never changes again. MNIST and FashionMNIST commit on the first sweep for almost every image (95th percentile 2 sweeps); Sudoku boards need a median of 3 sweeps, and the hardest 5% take 11 or more; (d) The loop starts from the zero state and, after each sweep, we record the distance to the state it finally settles in, relative to the size of that state, median over 1000 test samples. In MNIST and FashionMNIST, the model converges in about 20 sweeps; Sudoku, less contractive, needs about 60.

4.2 Empirical validation of the theory

Theorems 1 and 2 hold under the conditions of the models exactly reaching the equilibrium and fixed points, and under the assumption that $\beta \to 0$. As both of these assumptions are not realistic when it comes to numerical simulations, we now empirically check whether we can observe similar behavior in practice. First, we check whether the relaxation process reaches a fixed equilibrium, and then whether the EP update is a faithful estimate of both the BPTT and DEQ gradient. To conclude, we also show that inference on our circular model behaves as effective depth, where a subset of the test examples need more iterations to converge to the correct solutions. As datasets, we consider MNIST, FashionMNIST, and a set of 4×4 Sudoku boards.

Gradient Results. We initialize a recursive model with a single embedding layer, and two value nodes of width 128, and from the same weights we perform parameter updates of BPTT, PC with centered nudging, and using the IFT. The results, reported in Fig. 2(a,b), show that (i) BPTT matches the EP gradient only once it unrolls enough sweeps to reach the fixed point. The nudge value is also important: smaller values of β provide better approximation; and (ii) PC well approximates the IFT gradients, with better approximations again given by smaller values of β.

Computing the free phase. Since the free equilibrium is unique (Theorem 1), any procedure that drives the prediction errors to zero reaches the same state. We consider two: the *Energy relaxation* lets every value node reduce its errors in parallel, using only local information. This is the standard one defined in the background section for predictive coding models. Alternatively, we can continuously perform forward passes through the model. Here, we call a *sweep* a single application of $T_{\theta,\mathbf{o}}$. Sweeps are faster, but they only move information forward. In general, we compute the free phase using the PC formulation, but in some analyses it can be interesting to observe the number of forward passes needed to converge to the correct solution, as it coincides with the effective depth of the model. This is the method we use now.

As all of the trained models have spectral norm $\rho(J) < 1$, Theorem 1 predicts convergence to a unique state. Fig. 2(d) confirms this: the relative distance to the fixed point decays on every task until convergence. The vision models settle at $\rho(J) \approx 0.5$ and reach 10^{-4} in about 10 sweeps, while Sudoku trains to $\rho(J) = 0.85$ and needs about 40 iterations to reach convergence. Furthermore, to claim test time computation, the time at which its prediction becomes correct should depend on the input. This is what we observe in Fig. 2(c): for MNIST and FashionMNIST, the median input commits after one sweep, but harder to classify digits need more than 4 sweeps. On Sudoku, the median board needs 3 sweeps and the hardest 5% need 11 or more.

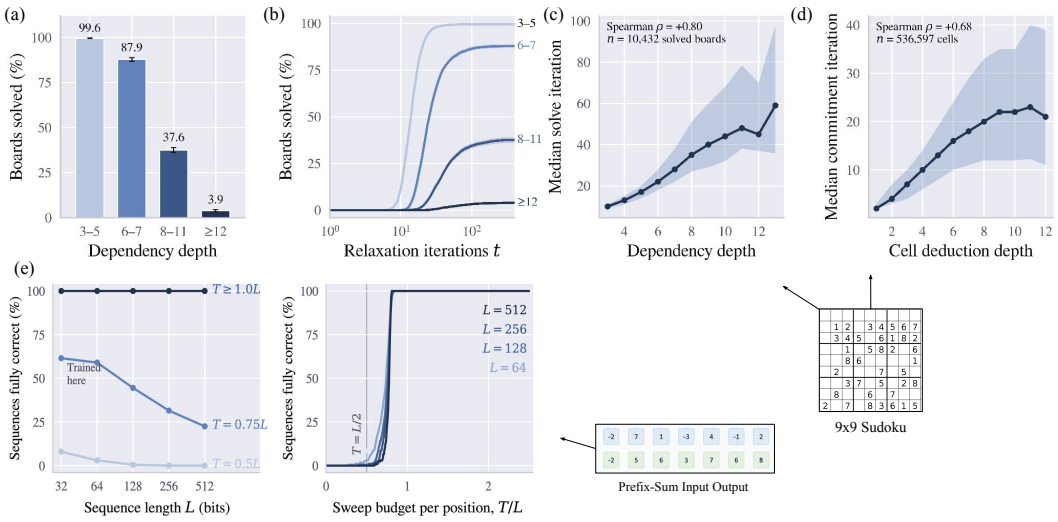


Figure 3: (a) fraction of boards solved per depth group; (b) fraction of boards solved within t relaxation iterations; deeper boards need larger budgets; (c) median solve iteration against depth (Spearman $\rho = +0.80$), among solved boards; (d) median iteration at which a cell becomes permanently correct against its deduction depth ($\rho = +0.68$): cells unlocked by deeper deductions are resolved later; (e) OOD sequence accuracy on the prefix-sum dataset, over the number of steps t.

5 Experiments

The goal of this section is to test our hypothesis that circular PC models are able to perform test-time computation, and solve puzzles that require a larger number of reasoning steps than the effective depth of the model. To do so, we consider the sudoku dataset Palm et al. (2018), and prefix-sum Anil et al. (2022). We also test image classification performance, and NLP.

Sudokus. We consider a circular model consisting of a token embedding layer and a single MLP-Mixer block Tolstikhin et al. (2021). This yields a circular model with 4 maps $g_{i,j}$, one per sub-layer of the MLP-Mixer: token and channel mixing, each followed by a feedforward map and normalization, which preserves the locality of every computation. We use GELU activations, a non-affine LayerNorm, and no residual connections, with a channel width of 256 and token-/channel-mixing hidden sizes of 162 and 512 ($\sim$302k parameters). Every map is kept contractive by capping its spectral norm at 0.7 after each update. We train on 180k puzzles with 17 to 34 given cells, augmented with the symmetry group of Palm et al. (2018), and test on boards of different dependency depths, i.e., the length of the longest deduction chain in the puzzle. Fig. 3(a–d) shows that harder puzzles are solved less frequently and require progressively more relaxation iterations. In particular, the solve iteration increases strongly with dependency depth ($\rho = 0.80$), and cells requiring deeper deductions are correctly committed later in the relaxation ($\rho = 0.68$). This confirms our hypothesis that more complex deductions require more computation.

Prefix-sum. This dataset tests whether recursive computation enables OOD generalization: we train on sequences of length 32 and test on lengths $L \in \{32, 64, 128, 256, 512\}$. Fig. 3(e) shows that circular PC generalizes to sequences substantially longer than those seen during training, reaching 100% test accuracy at every tested length when given enough computation ($T \geq L$). Conversely, performance deteriorates with sequence length when the budget grows more slowly than L, showing that additional test-time computation enables generalization to larger problem sizes. We use a circular loop of length 4 as for Sudokus, but replace the dense token-mixing matrices with symmetric 1-D convolutions of kernel size 3, so that the same weights apply to sequences of any length. Since no positional embedding is used, information can only travel along the sequence through local interactions, one position per convolution. The model has a channel width of 64 ($\sim$132k parameters) and is trained on $8{,}000$ sequences.

Table 1: Test accuracy on CIFAR-10/100. The experiments with a similar number of parameters have been re-run by us. The ones on the ResNets have been taken from the original work Qi et al. (2025).

	Similar number of parameters						ResNet-18	
Dataset	CPC	DualCPC	EP	PC	PC-CN	iPC	PC + FU	iPC + S
CIFAR-10	91.47 ± 0.71	$\mathbf{91.80} \pm 0.25$	90.85 ± 0.21	86.93 ± 0.29	88.41 ± 0.18	85.62 ± 0.35	92.39 ± 0.04	$\mathbf{92.47} \pm 0.01$
CIFAR-100 (top1)	69.35 ± 0.12	$\mathbf{71.70} \pm 0.26$	65.31 ± 0.47	62.91 ± 0.73	63.99 ± 0.11	61.48 ± 0.58	70.38 ± 0.20	70.90 ± 0.18
CIFAR-100 (top5)	87.22 ± 0.18	$\mathbf{92.66} \pm 0.10$	87.14 ± 0.25	82.13 ± 0.51	84.11 ± 0.16	80.56 ± 0.43	90.74 ± 0.10	90.47 ± 0.11

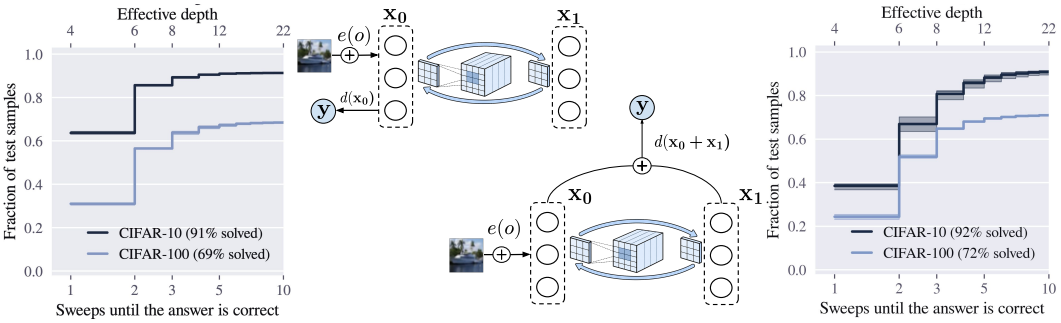


Figure 4: Left: The effective depth of our convolutional model trained on CIFAR10 and CIFAR100. Right: Effective depth of the model with the dual readout. In the middle, a graphical representation of the two convolutional loops, the standard one (left), and the one with double readout (right). Effective depth is computed as $3 + 2t$, where t is the number of sweeps.

5.1 Image Classification

Research in recent years has shown that deeper models tend to outperform shallow ones He et al. (2016). Predictive coding networks, however, have been shown not to be robust to this kind of scaling, as they tend to perform worse once depth exceeds 7 Pinchetti et al. (2025). In this section, we show that a recursive model with 5 convolutional layers, only two of which are recursive, exploits its effective depth to match state-of-the-art PC models trained on ResNet18s, despite having only $\sim$1.4M parameters. We compare against two baselines: VGG5 convolutional models with the same number of parameters, which we train ourselves, and the aforementioned PC variants for deep networks, for which we report the ResNet18 results stated in the original works. Results are in Table 1.

Set up. The model is a circular convolutional network with two value nodes, each a 256-channel 8×8 feature map, and five convolutional layers: Three input layers that map the image to two recurrent layers forming the loop, and a linear readout applied to the pooled state. Only the two layers inside the loop are recursive, so a single sweep corresponds to a network of depth 5, and t sweeps to an effective depth of $3 + 2t$. Both datasets use standard augmentation (random crops and horizontal flips), and the same architecture is used for CIFAR10 and CIFAR100. We also test a dual-readout formulation Figure 4, where we feed the decoder layer the sum $\mathbf{x_1} + \mathbf{x_0}$ of the two recursive nodes, instead of $\mathbf{x_0}$ alone. This adds no parameters, but the nudging signal now acts on both nodes simultaneously: the target-driven perturbation enters the loop at two points instead of one, and reaches every layer within a single sweep rather than after a full round trip.

Results. The circular model outperforms all the VGG5 baselines, and is on par with PC variants designed for ResNet18. The dual readout accounts for a consistent part of this gain, as it outperforms the single-readout model on CIFAR10 by almost half a percentage point, and CIFAR100 by 5% on its Top-5 accuracy. Fig. 4 shows the sweep at which each test image is permanently classified correctly: the median image commits after 2 sweeps (effective depth 7), while the hardest 5% need 5–6 sweeps (depth 13–15), so the model spends more computation on harder inputs. We attribute the effect of the dual readout to two factors. First, averaging two nodes that agree at equilibrium but differ along the relaxation trajectory acts as an ensemble over depth, reducing the variance of the prediction. Second, and more importantly, a nudge applied to both nodes gives every recursive layer a direct, one-step supervision signal, so the contrastive weight update is less attenuated by the contraction of the loop, which we observe as faster training and a higher final accuracy.

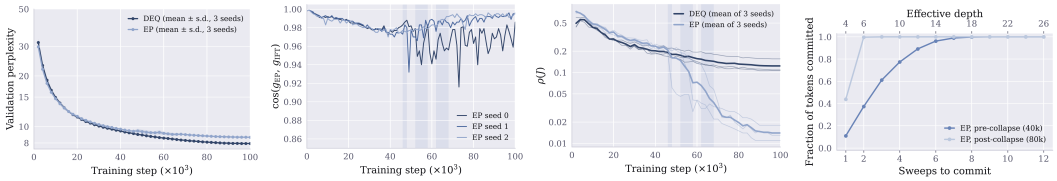


Figure 5: Transformer on TinyStories, EP versus DEQ. From left to right: validation perplexity over training EP gradient fidelity, $\cos(g_{\mathrm{EP}}, g_{\mathrm{IFT}})$, with the EP collapse windows shaded; spectral radius $\rho(J)$ of the loop Jacobian at the fixed point, same windows shaded; fraction of validation tokens whose prediction has committed (per-token loss within 0.01 nats of its converged value) after t sweeps, EP before (40k steps) and after (80k) the collapse, with the effective depth on the top axis.

5.2 Language Modeling

Table 2: PPL on TinyStories.

Method	Params	Test ppl ↓
DEQ	5.56M	$\mathbf{7.94} \pm 0.02$
CPC	5.56M	8.65 ± 0.08

Transformer models are the dominant architecture of modern natural language modelling, yet they remain almost absent from the predictive coding literature. To the best of our knowledge, existing results are limited to small models trained on small subsets of the One Billion Word benchmark (Chelba et al., 2014; Pinchetti et al., 2022; Salvatori et al., 2023b). In this section, we take a step towards closing this gap, and train a CPC model with a single recursive transformer layer on the TinyStories dataset (Eldan & Li, 2023).

Set up. Text is tokenized with a BPE vocabulary of 8192 tokens. The model is a autoregressive transformer block (Vaswani et al., 2017), formed by self-attention followed by a feedforward map, closed into a loop. It has two value nodes: one after the attention sub-layer, and one after the feedforward sub-layer, which also feeds the linear readout. The model has a width of 384, 6 attention heads, a context of 128 tokens, and $\sim$5.6M parameters. As in the previous sections, every map is kept contractive by capping its spectral norm after each update; in addition, we bound the gain of the LayerNorm inside the loop, and normalize queries and keys so that attention logits remain bounded. Each model is trained for 100k steps. As a baseline, we train a DEQ model with the same structure. Perplexity is measured on a set of 102k test tokens.

Linear-response nudging. Instead of relaxing the nudged energy F_β until convergence, we compute the response of the free equilibrium to the nudge directly. Since β is small, this response is well described by a linear approximation around the free equilibrium, and can be obtained by solving a linear system defined by the Hessian of the energy. We solve this system with the heavy-ball method (Polyak, 1964), which only requires Hessian–vector products. Here, the step size and momentum of the method are set by two global scalars, estimates of the extreme eigenvalues of the Hessian. The parameters are then updated using centered nudging.

Results. Trained with implicit gradients, the model reaches a test perplexity of 7.94 ± 0.02; trained with EP, it reaches 8.67 ± 0.08 (Table 2). The learning curves in figure 5 show that this gap is not present from the start: EP slightly leads during the first phase of training, and the two curves cross between 8k and 14k steps, after which the DEQ pulls ahead. The origin of the gap lies in the training trajectory. Midway through training, the EP-trained model undergoes a sudden drop, after which the loop map becomes strongly contractive and the model behaves almost as a feedforward network. The DEQ never undergoes this transition. In figure 5 we can observe how both gradient fidelity and radius of the Jacobian collapse at step 40k.

6 Conclusion

We showed that a cyclic energy-based network can realize the equilibrium computation of an implicit recurrent model, while EP provides a local equilibrium-based mechanism for training that computation without differentiating through its relaxation dynamics. This yields a model that uses relaxation time as a computational resource on Sudoku and prefix sums, while remaining competitive on image classification tasks. The practical interest of the construction lies in physical implementations: as depth is unrolled in time rather than in space, a fixed-size analog loop could in principle spend more time settling on harder inputs using a fixed-size array, while learning reduces to a local contrast between two measured steady states Kendall et al. (2020); Dillavou et al. (2022); Yi et al. (2023).

Reproducibility Statement

We provide a comprehensive description of the model and the experimental setup in the main paper; we also provide appendices for all training details. We will release the code in case of acceptance.

Ethics Statement

Bio-plausible machine learning is a scientific field that at the moment does not pose any ethical concern, if not that of fundamental progress in the field of AI and machine learning.

AI Usage

We have used LLMs for checking and polishing writing, for help with coding all the experiments in the paper. We reviewed all AI-assisted outputs and took steps to verify their correctness and reliability. We have used LLMs for aiding, verify, and polishing our mathematical statements and proofs. We however take full responsability for eventual mistakes.

References

Garrett E Alexander, Mahlon R DeLong, and Peter L Strick. Parallel organization of functionally segregated circuits linking basal ganglia and cortex. *Annual review of neuroscience*, 9(1):357–381, 1986.

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J Zico Kolter, and Roger B Grosse. Path independent equilibrium models can better exploit test-time computation. *Advances in Neural Information Processing Systems*, 35:7796–7809, 2022.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. Curran Associates, Inc., 2019.

Ciprian Chelba, Tomás Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*, 2014.

Sam Dillavou, Menachem Stern, Andrea J Liu, and Douglas J Durian. Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18(1):014040, 2022.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *CoRR*, abs/2305.07759, 2023.

Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1456), 2005.

Joshua I Gold and Michael N Shadlen. The neural basis of decision making. *Annu. Rev. Neurosci.*, 30(1):535–574, 2007.

Myoung Hoon Ha, Hyunjun Kim, Yoondo Sung, Youngha Jo, Min Kang, and Sang Wan Lee. Stable and scalable deep predictive coding networks with meta-prediction errors. In *International Conference on Learning Representations*, volume 2026, pp. 29268–29304, 2026.

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

Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B Issa, and James J DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature neuroscience*, 22(6):974–983, 2019.

Roberta M Kelly and Peter L Strick. Cerebellar loops with motor cortex and prefrontal cortex of a nonhuman primate. *The Journal of neuroscience*, 23(23):8432–8444, 2003.

Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*, 2020.

Tugdual Kerjan, Rasmus Høier, and Benjamin Scellier. Training a predictive coding network on imagenet using equilibrium propagation. *arXiv preprint arXiv:2606.03584*, 2026.

Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:633674, 2021.

Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv:2006.04182*, 2020.

Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022.

Ali Momeni, Babak Rahmani, Benjamin Scellier, Logan G Wright, Peter L McMahon, Clara C Wanjura, Yuhang Li, Anas Skalli, Natalia G Berloff, Tatsuhiro Onodera, et al. Training of physical neural networks. *Nature*, 645(8079):53–61, 2025.

Gaspard Oliviers, Mufeng Tang, and Rafal Bogacz. Bidirectional predictive coding. In *International Conference on Learning Representations*, volume 2026, pp. 59585–59619, 2026.

Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. *Advances in neural information processing systems*, 31, 2018.

Luca Pinchetti, Tommaso Salvatori, Yordan Yordanov, Beren Millidge, Yuhang Song, and Thomas Lukasiewicz. Predictive coding beyond gaussian distributions. *arXiv preprint arXiv:2211.03481*, 2022.

Luca Pinchetti, Chang Qi, Oleh Lokshyn, Cornelius Emde, Amine M'Charrak, Mufeng Tang, Simon Frieder, Bayar Menzat, Gaspard Oliviers, Rafal Bogacz, et al. Benchmarking predictive coding networks–made simple. In *International Conference on Learning Representations*, volume 2025, pp. 35701–35734, 2025.

B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, pp. 1–17, 1964.

Chang Qi, Matteo Forasassi, Thomas Lukasiewicz, and Tommaso Salvatori. Towards the training of deeper predictive coding neural networks. *arXiv preprint arXiv:2506.23800*, 2025.

Rajesh P. N. 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.

Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *arXiv:2201.13180*, 2022a.

Tommaso Salvatori, Yuhang Song, Zhenghua Xu, Thomas Lukasiewicz, and Rafal Bogacz. Reverse differentiation via predictive coding. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. AAAI Press, 2022b.

Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870*, 2023a.

Tommaso Salvatori, Luca Pinchetti, Amine M'Charrak, Beren Millidge, and Thomas Lukasiewicz. Predictive coding beyond correlations. *arXiv preprint arXiv:2306.15479*, 2023b.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017.

Benjamin Scellier, Maxence Ernoult, Jack Kendall, and Suhas Kumar. Energy-based learning algorithms for analog computing: a comparative study. *Advances in Neural Information Processing Systems*, 36, 2024.

Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

Jeffrey Seely. Sheaf cohomology of linear predictive coding networks. *arXiv preprint arXiv:2511.11092*, 2025.

Jeffrey Seely and Julian Gould. Augmented lagrangian predictive coding. *arXiv preprint arXiv:2605.31022*, 2026.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

James C. R. 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), 2017.

Logan G Wright, Tatsuhiro Onodera, Martin M Stein, Tianyu Wang, Darren T Schachter, Zoey Hu, and Peter L McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601 (7894):549–555, 2022.

Su-in Yi, Jack D Kendall, R Stanley Williams, and Suhas Kumar. Activity-difference training of deep neural networks using memristor crossbars. *Nature Electronics*, 6(1):45–51, 2023.

Nicolas Zucchet and Joao Sacramento. Beyond backpropagation: bilevel optimization through implicit differentiation and equilibrium propagation. *Neural Computation*, 34(12):2309–2346, 2022.

A Formal statements and proofs

This appendix contains the precise versions of the two statements of the main body, together with their proofs.

A.1 Setting

The graph and its state. The vertex set is $\mathcal{V} = \{\mathrm{in}, 1, \dots, L, \mathrm{out}\}$. The vertices $1, \dots, L$ form the loop and carry value nodes $\mathbf{x}_i \in \mathbb{R}^{d_i}$; the input vertex is clamped to the observation $\mathbf{o}$ and is therefore not a variable, and the output vertex carries $\mathbf{x}_{\mathrm{out}} \in \mathbb{R}^{d_{\mathrm{out}}}$. To lighten Eq. 6 we collect the edge maps of the loop into

$$\phi_1 \coloneqq g_{L,1}, \qquad \phi_i \coloneqq g_{i-1,i} \quad (2 \le i \le L-1), \qquad \phi_L \coloneqq g_{L-1,L}\big(\,\cdot\,, e_\theta(\mathbf{o})\big), \tag{8}$$

so that $\phi_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ for every i, under the cyclic convention $d_0 \coloneqq d_L$; the dependence of ϕ_L on θ and $\mathbf{o}$ is left implicit. We write

$$\mathbf{x}_{\mathrm{loop}} = (\mathbf{x}_1, \dots, \mathbf{x}_L) \in \mathbb{R}^n, \quad n = \sum_{i=1}^{L} d_i, \qquad \mathbf{x} = (\mathbf{x}_{\mathrm{loop}}, \mathbf{x}_{\mathrm{out}}) \in \mathbb{R}^{n+d_{\mathrm{out}}} \tag{9}$$

for the state of the loop and for the vector of all free value nodes.

The energy. With the predictions of Eq. 6, the energy of Eq. 2 splits into a loop part and a single decoder term,

$$E(\theta, \mathbf{x}) = \underbrace{\frac{1}{2}\sum_{i=1}^{L}\big\|\mathbf{x}_i - \hat{\mathbf{x}}_i\big\|_2^2}_{=:\, E_{\mathrm{loop}}(\theta, \mathbf{x}_{\mathrm{loop}})} + \frac{1}{2}\big\|\mathbf{x}_{\mathrm{out}} - d_\theta(\mathbf{x}_L)\big\|_2^2, \tag{10}$$

the first group involving $\mathbf{x}_{\mathrm{out}}$ through no term at all.[2]

The two maps. Because all value nodes are updated at the same time, the object attached to the loop by the inference dynamics is the *simultaneous prediction map*

$$S_{\theta,\mathbf{o}} : \mathbb{R}^n \to \mathbb{R}^n, \qquad S_{\theta,\mathbf{o}}(\mathbf{x}_{\mathrm{loop}}) = \big(\phi_1(\mathbf{x}_L),\, \phi_2(\mathbf{x}_1),\, \ldots,\, \phi_L(\mathbf{x}_{L-1})\big), \tag{11}$$

which stacks every local prediction evaluated at the current state, whereas the deep equilibrium model of Section 3.1 iterates the composition around the loop,

$$T_{\theta,\mathbf{o}} : \mathbb{R}^{d_L} \to \mathbb{R}^{d_L}, \qquad T_{\theta,\mathbf{o}} = \phi_L \circ \phi_{L-1} \circ \cdots \circ \phi_1. \tag{12}$$

The first acts on the joint state, the second on the main value node alone; Lemma 3 relates them. Since the i-th block of $\mathbf{x}_{\mathrm{loop}} - S_{\theta,\mathbf{o}}(\mathbf{x}_{\mathrm{loop}})$ is the prediction error $\boldsymbol{\varepsilon}_i$, the loop energy is exactly the squared residual of the simultaneous map,

$$E_{\mathrm{loop}}(\theta, \mathbf{x}_{\mathrm{loop}}) = \frac{1}{2}\big\|\mathbf{r}(\mathbf{x}_{\mathrm{loop}})\big\|_2^2, \qquad \mathbf{r}(\mathbf{x}_{\mathrm{loop}}) \coloneqq \mathbf{x}_{\mathrm{loop}} - S_{\theta,\mathbf{o}}(\mathbf{x}_{\mathrm{loop}}). \tag{13}$$

This identity is what the whole of Section A.2 rests on.

Remark 1 (The parameter gradient carries the encoder) *Throughout, $\nabla_\theta E(\theta, \mathbf{x})$ denotes the gradient of the map $\theta \mapsto E(\theta, \mathbf{x})$ at fixed state and fixed observation $\mathbf{o}$. By Eq. 8 this includes the path through the injection $e_\theta(\mathbf{o})$, which enters $\hat{\mathbf{x}}_L$ and hence $\boldsymbol{\varepsilon}_L$. Treating that injection as a precomputed constant would differentiate a different function, one in which the encoder parameters do not appear at all; Theorem 4 then applies to that other function and says nothing about e_θ.*

Assumption Every map of Eq. 8 is continuously differentiable and q-Lipschitz with respect to the Euclidean norm, for a common modulus $q < 1$, the map ϕ_L being required to be so in its first argument at fixed $\mathbf{o}$. The decoder d_θ is continuously differentiable, and no contractivity is asked of it.

Three lemmas precede the results. The first eliminates the vertices outside the loop, the second turns Assumption A.1 into a bound on the joint map, and the third identifies the fixed points of the two maps of Eqs. 11 and 12.

Lemma 1 (Elimination of the output vertex) *A state $\mathbf{x} = (\mathbf{x}_{\mathrm{loop}}, \mathbf{x}_{\mathrm{out}})$ is a stationary point of $E(\theta, \cdot)$ in the free value nodes if and only if*

$$\nabla_{\mathbf{x}_{\mathrm{loop}}} E_{\mathrm{loop}}(\theta, \mathbf{x}_{\mathrm{loop}}) = 0 \qquad \textit{and} \qquad \mathbf{x}_{\mathrm{out}} = d_\theta(\mathbf{x}_L), \tag{14}$$

and at any such state $E(\theta, \mathbf{x}) = E_{\mathrm{loop}}(\theta, \mathbf{x}_{\mathrm{loop}})$.

Proof. By Eq. 10, stationarity in $\mathbf{x}_{\mathrm{out}}$ reads $\boldsymbol{\varepsilon}_{\mathrm{out}} = \mathbf{x}_{\mathrm{out}} - d_\theta(\mathbf{x}_L) = 0$. The output vertex has no children, so it enters no prediction, and the only place it touches the remaining stationarity conditions is the term $-\mathbf{J}_{d_\theta}(\mathbf{x}_L)^\top \boldsymbol{\varepsilon}_{\mathrm{out}}$ of $\partial E/\partial \mathbf{x}_L$, which vanishes together with $\boldsymbol{\varepsilon}_{\mathrm{out}}$. The conditions on the loop variables reduce accordingly to those of E_{loop}. The converse substitution is immediate, as is the value of the energy, the decoder term being zero. □

Lemma 2 (Contraction of the simultaneous map) *Under Assumption A.1, the map $S_{\theta,\mathbf{o}}$ is q-Lipschitz on $\mathbb{R}^n$, and $\|\mathbf{J}_S(\mathbf{x}_{\mathrm{loop}})\|_2 \le q$ at every state.*

[2] The input vertex has no parents, so its prediction is an empty sum and its error term is $\frac{1}{2}\|\mathbf{o}\|_2^2$, a constant independent of both the free state and the parameters. We discard it throughout, which affects no stationarity condition and no parameter gradient.

Proof. Let P be the coordinate permutation of $\mathbb{R}^n$ sending $(\mathbf{x}_1, \dots, \mathbf{x}_L)$ to $(\mathbf{x}_L, \mathbf{x}_1, \dots, \mathbf{x}_{L-1})$, so that the i-th block of $P\mathbf{x}_{\text{loop}}$ is $\mathbf{x}_{i-1}$ and lies in $\mathbb{R}^{d_{i-1}}$, the domain of ϕ_i. Letting $\boldsymbol{\phi}$ act blockwise on that decomposition, $\boldsymbol{\phi}(\mathbf{u}) = (\phi_1(\mathbf{u}_1), \dots, \phi_L(\mathbf{u}_L))$, a comparison with Eq. 11 gives the factorization

$$S_{\theta,\mathbf{o}} = \boldsymbol{\phi} \circ P. \tag{15}$$

For any $\mathbf{u}, \mathbf{u}'$,

$$\big\|\boldsymbol{\phi}(\mathbf{u}) - \boldsymbol{\phi}(\mathbf{u}')\big\|_2^2 = \sum_{i=1}^{L} \big\|\phi_i(\mathbf{u}_i) - \phi_i(\mathbf{u}_i')\big\|_2^2 \le q^2 \sum_{i=1}^{L} \|\mathbf{u}_i - \mathbf{u}_i'\|_2^2 = q^2 \|\mathbf{u} - \mathbf{u}'\|_2^2, \tag{16}$$

so $\boldsymbol{\phi}$ is q-Lipschitz, and P is orthogonal; the first claim follows from Eq. 15. Differentiating that same factorization gives $\mathbf{J}_S = \mathbf{J}_{\boldsymbol{\phi}} P$ with $\mathbf{J}_{\boldsymbol{\phi}} = \operatorname{blockdiag}(\mathbf{J}_{\phi_1}, \dots, \mathbf{J}_{\phi_L})$, whence

$$\|\mathbf{J}_S\|_2 \le \|\mathbf{J}_{\boldsymbol{\phi}}\|_2 \, \|P\|_2 = \max_{1 \le i \le L} \|\mathbf{J}_{\phi_i}\|_2 \le q, \tag{17}$$

the last bound holding pointwise because a differentiable q-Lipschitz map has Jacobian of operator norm at most q. □

Lemma 3 (The two maps have the same fixed points) *The assignment $\mathbf{x}_{\text{loop}} \mapsto \mathbf{x}_L$ is a bijection from the fixed points of $S_{\theta,\mathbf{o}}$ onto those of $T_{\theta,\mathbf{o}}$, with inverse $\mathbf{s} \mapsto \big(\phi_1(\mathbf{s}),\, \phi_2(\phi_1(\mathbf{s})),\, \dots,\, \mathbf{s}\big)$.*

Proof. If $S_{\theta,\mathbf{o}}(\mathbf{x}_{\text{loop}}) = \mathbf{x}_{\text{loop}}$ then $\mathbf{x}_1 = \phi_1(\mathbf{x}_L)$ and $\mathbf{x}_i = \phi_i(\mathbf{x}_{i-1})$ for $2 \le i \le L$; substituting the first $L-1$ identities into the last one gives $\mathbf{x}_L = (\phi_L \circ \cdots \circ \phi_1)(\mathbf{x}_L) = T_{\theta,\mathbf{o}}(\mathbf{x}_L)$. Conversely, if $T_{\theta,\mathbf{o}}(\mathbf{s}) = \mathbf{s}$, the state defined by $\mathbf{x}_1 = \phi_1(\mathbf{s})$, by $\mathbf{x}_i = \phi_i(\mathbf{x}_{i-1})$ for $2 \le i \le L-1$, and by $\mathbf{x}_L = \mathbf{s}$, satisfies every block of the fixed-point condition of $S_{\theta,\mathbf{o}}$, the last one precisely because $\mathbf{s}$ is fixed by the composition. The two constructions are mutually inverse. □

A.2 Inference

Theorem 3 (Formal version of Theorem 1) *Let Assumption A.1 hold. Then $E(\theta, \cdot)$ has exactly one stationary point $\mathbf{x}^\star$ in the free value nodes, and*

- *(i) $E(\theta, \mathbf{x}^\star) = 0$, and $\mathbf{x}^\star$ is the unique global minimizer of $E(\theta, \cdot)$;*
- *(ii) $\mathbf{x}_L^\star$ is the unique fixed point of $T_{\theta,\mathbf{o}}$, and the remaining value nodes are obtained from it by a single pass around the loop, namely $\mathbf{x}_1^\star = \phi_1(\mathbf{x}_L^\star)$ and $\mathbf{x}_i^\star = \phi_i(\mathbf{x}_{i-1}^\star)$ for $2 \le i \le L-1$, together with $\mathbf{x}_{\text{out}}^\star = d_\theta(\mathbf{x}_L^\star)$;*
- *(iii) every trajectory of the gradient flow $\dot{\mathbf{x}} = -\nabla_{\mathbf{x}} E(\theta, \mathbf{x})$ that converges to a stationary point converges to $\mathbf{x}^\star$.*

Proof. By Lemma 1 it suffices to prove the corresponding statements for E_{loop}, the output value node being slaved to $\mathbf{x}_L$.

Existence, and vanishing energy. By Lemma 2 the map $S_{\theta,\mathbf{o}}$ is a contraction of the complete space $\mathbb{R}^n$, so the Banach fixed-point theorem provides a unique $\mathbf{x}_{\text{loop}}^\star$ with $S_{\theta,\mathbf{o}}(\mathbf{x}_{\text{loop}}^\star) = \mathbf{x}_{\text{loop}}^\star$. There $\mathbf{r} = 0$, hence $E_{\text{loop}} = 0$ by Eq. 13, and since the energy is nonnegative everywhere this state is a global minimizer; it is the only one, a squared norm vanishing exactly where its argument does.

Uniqueness of the stationary point. Differentiating Eq. 13,

$$\nabla_{\mathbf{x}_{\text{loop}}} E_{\text{loop}}(\theta, \mathbf{x}_{\text{loop}}) = \big(\mathbf{I} - \mathbf{J}_S(\mathbf{x}_{\text{loop}})\big)^\top \mathbf{r}(\mathbf{x}_{\text{loop}}). \tag{18}$$

By Lemma 2 we have $\|\mathbf{J}_S\|_2 \le q < 1$ at every state, so $\mathbf{I} - \mathbf{J}_S$ is invertible everywhere, its inverse being the convergent Neumann series $\sum_{k \ge 0} \mathbf{J}_S^k$. Hence Eq. 18 vanishes if and only if $\mathbf{r}(\mathbf{x}_{\text{loop}}) = 0$, that is, exactly at the fixed point just constructed. This proves both the uniqueness and item (i).

Identification with the deep equilibrium model. Item (ii) is Lemma 3 applied to $\mathbf{x}_{\text{loop}}^\star$, the fixed point of $T_{\theta,\mathbf{o}}$ being unique because that of $S_{\theta,\mathbf{o}}$ is and the correspondence is bijective, together with Lemma 1 for the output vertex.

Gradient flow. A convergent trajectory has a stationary limit, and there is only one, which is item (iii). □

The next two statements sharpen the theorem in the two directions one would want, and both come essentially for free from the residual form of the energy: the first strengthens the convergence of the free phase from conditional to global, the second shows that the equilibrium so reached is nondegenerate, which is precisely the hypothesis under which the learning result is proved.

Proposition 1 (Quadratic growth and global convergence) *Under Assumption A.1,*

$$E_{\mathrm{loop}}(\theta, \mathbf{x}_{\mathrm{loop}}) \;\geq\; \frac{(1-q)^2}{2}\big\|\mathbf{x}_{\mathrm{loop}} - \mathbf{x}^\star_{\mathrm{loop}}\big\|_2^2 \qquad \textit{for every } \mathbf{x}_{\mathrm{loop}} \in \mathbb{R}^n, \tag{19}$$

and $E(\theta, \cdot)$ has bounded sublevel sets. If moreover the maps of Eq. 8 and the decoder are twice continuously differentiable, so that $\nabla_{\mathbf{x}} E$ is locally Lipschitz, then the gradient flow is defined for all $t \geq 0$ from any initial state and converges to $\mathbf{x}^\star$.

Proof. Using $\mathbf{x}^\star_{\mathrm{loop}} = S_{\theta,\mathbf{o}}(\mathbf{x}^\star_{\mathrm{loop}})$ and Lemma 2,

$$\big\|\mathbf{r}(\mathbf{x}_{\mathrm{loop}})\big\|_2 = \Big\|(\mathbf{x}_{\mathrm{loop}} - \mathbf{x}^\star_{\mathrm{loop}}) - \big(S_{\theta,\mathbf{o}}(\mathbf{x}_{\mathrm{loop}}) - S_{\theta,\mathbf{o}}(\mathbf{x}^\star_{\mathrm{loop}})\big)\Big\|_2 \tag{20}$$

$$\geq (1-q)\big\|\mathbf{x}_{\mathrm{loop}} - \mathbf{x}^\star_{\mathrm{loop}}\big\|_2, \tag{21}$$

which is Eq. 19 after squaring. Consequently the sublevel set $\{E \leq c\}$ confines $\mathbf{x}_{\mathrm{loop}}$ to a ball, on which the continuous map d_θ is bounded, and the decoder term of Eq. 10 then bounds $\mathbf{x}_{\mathrm{out}}$ through $\|\mathbf{x}_{\mathrm{out}} - d_\theta(\mathbf{x}_L)\|_2 \leq \sqrt{2c}$; the sublevel set is therefore bounded.

For the flow, $\frac{\mathrm{d}}{\mathrm{d}t}E = -\|\nabla_{\mathbf{x}} E\|_2^2 \leq 0$, so a trajectory remains in the compact sublevel set of its initial value, and local Lipschitz continuity of the gradient yields a solution for all $t \geq 0$. Its ω-limit set is then nonempty, compact and invariant, and the energy is constant on it, being the monotone limit of $E(\mathbf{x}(t))$; along the trajectory issued from any of its points the energy is therefore constant, forcing $\nabla_{\mathbf{x}} E = 0$ there. By Theorem 3 the ω-limit set is $\{\mathbf{x}^\star\}$, and the trajectory converges to it. □

Proposition 2 (Nondegeneracy of the free equilibrium) *Assume in addition to Assumption A.1 that the maps of Eq. 8 and the decoder are twice continuously differentiable. Then the Hessian $\mathbf{H} = \nabla^2_{\mathbf{xx}} E(\theta, \mathbf{x}^\star)$ in the free value nodes is positive definite.*

Proof. Write the full energy as $E = \frac{1}{2}\|\mathbf{R}(\mathbf{x})\|_2^2$ with $\mathbf{R}(\mathbf{x}) = \mathbf{x} - S^{\mathrm{full}}_{\theta,\mathbf{o}}(\mathbf{x})$, where $S^{\mathrm{full}}_{\theta,\mathbf{o}}$ appends the block $d_\theta(\mathbf{x}_L)$ to Eq. 11. Then

$$\mathbf{H} = \mathbf{J}_{\mathbf{R}}^\top \mathbf{J}_{\mathbf{R}} + \sum_a R_a \, \nabla^2 R_a, \tag{22}$$

and the second sum vanishes at $\mathbf{x}^\star$, where $\mathbf{R}(\mathbf{x}^\star) = 0$ by Theorem 3. It remains to show that $\mathbf{J}_{\mathbf{R}} = \mathbf{I} - \mathbf{J}_{S^{\mathrm{full}}}$ is invertible. No vertex reads $\mathbf{x}_{\mathrm{out}}$, so in the block ordering $(\mathbf{x}_{\mathrm{loop}}, \mathbf{x}_{\mathrm{out}})$,

$$\mathbf{I} - \mathbf{J}_{S^{\mathrm{full}}} = \begin{pmatrix} \mathbf{I} - \mathbf{J}_S & \mathbf{0} \\ -\mathbf{D} & \mathbf{I} \end{pmatrix}, \qquad \mathbf{D} = \frac{\partial\, d_\theta(\mathbf{x}_L)}{\partial \mathbf{x}_{\mathrm{loop}}}, \tag{23}$$

which is block lower triangular with invertible diagonal blocks, the upper one by Lemma 2. Hence $\mathbf{J}_{\mathbf{R}}$ is invertible and $\mathbf{H} = \mathbf{J}_{\mathbf{R}}^\top \mathbf{J}_{\mathbf{R}} \succ 0$. □

Two remarks are in order, and the first concerns where the hypothesis has been placed. Assumption A.1 constrains each edge map of the loop separately, and not the composition $T_{\theta,\mathbf{o}}$, which is the object the deep equilibrium model iterates and the one a reader would expect to see assumed contractive. The next remark shows that this is not a matter of convenience.

Remark 2 (Why the hypothesis is placed on the individual maps) *Contractivity of $T_{\theta,\mathbf{o}}$ alone secures the unique fixed point, but not the uniqueness of the stationary point of the energy, which is a statement about arbitrary states rather than about consistent ones. Solving $\big(\mathbf{I} - \mathbf{J}_S(\mathbf{x}_{\mathrm{loop}})\big)\mathbf{v} = 0$ blockwise gives $\mathbf{v}_1 = \mathbf{J}_{\phi_1}(\mathbf{x}_L)\mathbf{v}_L$ and $\mathbf{v}_i = \mathbf{J}_{\phi_i}(\mathbf{x}_{i-1})\mathbf{v}_{i-1}$, whence*

$$\mathbf{I} - \mathbf{J}_S(\mathbf{x}_{\mathrm{loop}}) \textit{ is singular} \iff \mathbf{I} - \mathbf{J}_{\phi_L}(\mathbf{x}_{L-1}) \cdots \mathbf{J}_{\phi_1}(\mathbf{x}_L) \textit{ is singular}, \tag{24}$$

a product of Jacobians evaluated at value nodes that bear no relation to one another unless the state is already a fixed point. Bounding the Jacobian of the composition along consistent states does not control that product, whereas Assumption A.1 bounds its norm by q^L uniformly. The assumption is thus strictly stronger than contractivity of $T_{\theta,\mathbf{o}}$, which it implies with modulus q^L.

Remark 3 (Scope) *Assumption A.1 guarantees convergence of the direct iteration* $\mathbf{x}_{\text{loop}}^{k+1} = S_{\theta,\mathbf{o}}(\mathbf{x}_{\text{loop}}^{k})$ *at rate* q*, and of* $\mathbf{s}^{k+1} = T_{\theta,\mathbf{o}}(\mathbf{s}^{k})$ *at rate* q^L*, but not that of an arbitrary accelerated fixed-point solver, nor that of a particular discretization of the gradient flow, which requires in addition a step size adapted to the curvature of the energy. The correspondence established above is one of terminal states and not of trajectories: these procedures traverse the state space differently, yet whenever one of them settles at a stationary solution of its own equilibrium problem, that solution is* $\mathbf{x}^\star$*.*

A.3 Learning

We now turn to the parameter update. Let $\mathbf{x}^0(\theta)$ denote a free equilibrium, $\nabla_{\mathbf{x}} E(\theta, \mathbf{x}^0) = 0$, and let

$$\mathcal{L}(\theta) = C\big(\mathbf{x}_{\text{out}}^0(\theta), \mathbf{y}\big) \tag{25}$$

be the supervised cost evaluated there, which is the quantity the implicit gradient differentiates. We regard C as a function of the whole free state, constant in every block but the output one, so that $\nabla_{\mathbf{x}} C$ is supported on the output vertex; this is the precise sense in which the nudge is a force applied at the output alone. For β in a neighborhood of the origin, the nudged equilibrium $\mathbf{x}^\beta$ satisfies $\nabla_{\mathbf{x}}\big[E(\theta, \mathbf{x}^\beta) + \beta\, C(\mathbf{x}^\beta, \mathbf{y})\big] = 0$, and all gradients in θ are understood as in Remark 1.

Theorem 4 (Formal version of Theorem 2) *Let* $\mathbf{x}^0$ *be a free equilibrium at which* $\mathbf{H} = \nabla_{\mathbf{xx}}^2 E(\theta, \mathbf{x}^0) \succ 0$*, let* E *be twice continuously differentiable jointly in* $(\theta, \mathbf{x})$ *on a neighborhood of* $(\theta, \mathbf{x}^0)$*, and let* C *be twice continuously differentiable in* $\mathbf{x}$ *on a neighborhood of* $\mathbf{x}^0$*. If both phases are solved exactly, then* $\beta \mapsto \mathbf{x}^\beta$ *is well defined and continuously differentiable near* $\beta = 0$*, and*

$$\lim_{\beta \to 0} G_{\text{EP}}(\beta) = \lim_{\beta \to 0} \frac{\nabla_\theta E(\theta, \mathbf{x}^\beta) - \nabla_\theta E(\theta, \mathbf{x}^0)}{\beta} = \nabla_\theta \mathcal{L}(\theta). \tag{26}$$

If moreover E *is* C^3 *jointly and* C *is* C^3 *in* $\mathbf{x}$*, then* $G_{\text{EP}}(\beta) = \nabla_\theta \mathcal{L}(\theta) + O(\beta)$*; and if* E *is* C^4 *jointly and* C *is* C^4 *in* $\mathbf{x}$*, the centered estimator satisfies*

$$G_{\text{cEP}}(\beta) \coloneqq \frac{\nabla_\theta E(\theta, \mathbf{x}^{+\beta}) - \nabla_\theta E(\theta, \mathbf{x}^{-\beta})}{2\beta} = \nabla_\theta \mathcal{L}(\theta) + O(\beta^2). \tag{27}$$

Proof. Step 1: the nudged branch. Define

$$\Phi(\mathbf{x}, \beta) \coloneqq \nabla_{\mathbf{x}} E(\theta, \mathbf{x}) + \beta\, \nabla_{\mathbf{x}} C(\mathbf{x}, \mathbf{y}), \tag{28}$$

continuously differentiable in $(\mathbf{x}, \beta)$ by hypothesis and vanishing at $(\mathbf{x}^0, 0)$. Its partial Jacobian in $\mathbf{x}$ at that point is $\mathbf{H}$, invertible by positive definiteness, so the implicit function theorem provides a neighborhood of $\beta = 0$ carrying a unique continuously differentiable branch $\beta \mapsto \mathbf{x}^\beta$ with $\Phi(\mathbf{x}^\beta, \beta) = 0$, equal to $\mathbf{x}^0$ at the origin. If E and C are C^{k+1}, then Φ is C^k and so is the branch.

Step 2: the response to the nudge. Differentiating $\Phi(\mathbf{x}^\beta, \beta) = 0$ with respect to β at $\beta = 0$,

$$\mathbf{H}\, \dot{\mathbf{x}}^0 + \nabla_{\mathbf{x}} C(\mathbf{x}^0, \mathbf{y}) = 0, \qquad \text{that is} \qquad \dot{\mathbf{x}}^0 \coloneqq \left.\frac{\mathrm{d}\mathbf{x}^\beta}{\mathrm{d}\beta}\right|_{\beta=0} = -\mathbf{H}^{-1} \nabla_{\mathbf{x}} C. \tag{29}$$

Step 3: the implicit gradient. Since $\nabla_{\mathbf{x}} E$ is continuously differentiable jointly and $\mathbf{H}$ is invertible, the same theorem applied to $\nabla_{\mathbf{x}} E(\theta, \mathbf{x}^0(\theta)) = 0$ in the variable θ yields a differentiable equilibrium branch with $\partial \mathbf{x}^0 / \partial \theta = -\mathbf{H}^{-1} \nabla_{\mathbf{x}\theta}^2 E$, the matrix $\nabla_{\mathbf{x}\theta}^2 E$ having entries $\partial^2 E / \partial x_a \partial \theta_b$. The chain rule and the symmetry of $\mathbf{H}$ then give

$$\nabla_\theta \mathcal{L} = \Big(\frac{\partial \mathbf{x}^0}{\partial \theta}\Big)^\top \nabla_{\mathbf{x}} C = -\big(\nabla_{\mathbf{x}\theta}^2 E\big)^\top \mathbf{H}^{-1} \nabla_{\mathbf{x}} C. \tag{30}$$

Step 4: the estimator. The numerator of $G_{\text{EP}}(\beta)$ is a difference quotient of $u(\beta) \coloneqq \nabla_\theta E(\theta, \mathbf{x}^\beta)$, differentiable at the origin by Step 1. By the chain rule and Eq. 29,

$$\lim_{\beta \to 0} G_{\text{EP}}(\beta) = u'(0) = \nabla_{\theta\mathbf{x}}^2 E\, \dot{\mathbf{x}}^0 = -\nabla_{\theta\mathbf{x}}^2 E\, \mathbf{H}^{-1} \nabla_{\mathbf{x}} C. \tag{31}$$

As E is twice continuously differentiable, the symmetry of second derivatives gives $\nabla^2_{\theta\mathbf{x}} E = (\nabla^2_{\mathbf{x}\theta} E)^\top$, so Eqs. 31 and 30 coincide, which proves Eq. 26.

Step 5: the orders. Under the stated additional smoothness, u is C^2, respectively C^3, by Step 1. Taylor expansion about the origin gives

$$u(\beta) - u(0) = \beta\, u'(0) + \tfrac{1}{2}\beta^2 u''(0) + o(\beta^2), \tag{32}$$

$$u(\beta) - u(-\beta) = 2\beta\, u'(0) + \tfrac{1}{3}\beta^3 u'''(0) + o(\beta^3), \tag{33}$$

the even-order terms cancelling by symmetry in the second line. Dividing by β and by 2β respectively yields $G_{\mathrm{EP}}(\beta) = u'(0) + O(\beta)$ and $G_{\mathrm{cEP}}(\beta) = u'(0) + O(\beta^2)$. □

B Experiments

All three tasks share one model family and one training loop: a weight-tied recurrent block expressed as a small predictive-coding circuit, in which every linear sub-layer is a value node carrying a local squared-residual energy, iterated to a joint equilibrium. Recurrent weights θ are updated with the EP contrastive rule. We keep every recurrent map contractive by capping its spectral norm after each optimizer step (Sedghi et al., 2018); this cap is the single knob governing the loop gain $\rho(J)$, and in every task below it is the hyperparameter to which final performance is most sensitive. All the hyperparameters we report below have been selected after extensive Optuna searches, and each reported metric has been evaluated on 5 different random seeds.

B.1 Sudoku

Model. We cast the Palm-2018 *sudoku-hard* benchmark as a closed 4-vode loop $s \to h_t \to t \to h_c \to s$, implemented as a single-block, attention-free MLP-Mixer: token-mixing across the 81 cells plays the role of constraint propagation, channel-mixing the role of per-cell digit inference. The loop's only normalization is a non-affine LayerNorm; there is no residual connection, and we use GELU throughout. The model has 301,952 parameters, a channel width of 256, and token-/channel-mixing hidden sizes of 162 and 512, respectively.

Data. We train on 180k puzzles, holding out 18k for validation and 18k for test, with givens drawn uniformly between 17 and 34. We apply the Palm-2018 symmetry group as augmentation at every training step.

Equilibrium and optimizer. We estimate the equilibrium with a centered, fused $\pm\beta$ estimator and solve it with a hybrid scheme: Anderson acceleration for the free phase, heavy-ball momentum for the nudged phase, run for 150 and 500 steps respectively. We use a spectral cap of 0.70, $\beta = 0.166$, and a variance-floor penalty of 2.92 targeting an equilibrium variance of 0.099. We optimize with AdamW (lr 1.05×10^{-3}, weight decay 1.21×10^{-4}), scaling the per-layer learning rate by 0.92 for token-mixing weights and 0.43 for channel-mixing weights, under a single warmup-then-cosine schedule. Each virtual epoch draws 8,000 fresh puzzles at batch size 256 (31 steps); we train for 200 epochs (6,200 steps total) and checkpoint on the best validation board accuracy, evaluated every two epochs.

B.2 Prefix sums (cumulative XOR)

Model. We port the Sudoku block to one dimension: the two dense token-mixing matrices become symmetric k=3 convolutions, so the same weights run at any sequence length. There is no residual connection and no positional embedding. The channel width is 64 (hidden size 256 for both mixers), for a total of 132,226 parameters. We train at length $L = 32$ on 8,000 examples (2,000 held out, fixed seed 42) and evaluate up to $L = 512$ with the weights unchanged.

Equilibrium and optimizer. Equilibrium is estimated with a centered, fused $\pm\beta$ estimator and solved by plain fixed-point sweeps, without acceleration, for 40 free-phase and 400 nudged-phase steps. Unlike the other two tasks, this equilibrium is *ungated*: we run truncated EP with no convergence-tolerance gate, and anneal the step budget from epoch 30 onward at $\times 0.9$ per epoch

down to a floor of 32 steps. We use a spectral cap of 1.05 (bounded via the FFT, since the maps are convolutional) with an additional per-step clamp of $\rho(J) \leq 0.8$ applied only downward as a rail, and $\beta = 0.294$. The variance-floor penalty is 2.28 at a target of 0.079. We optimize with AdamW (lr 1.36×10^{-3}, weight decay 1.99×10^{-4}), batch size 256, for 46 epochs from scratch (≈25 minutes on one H100). We keep the final iterate with no checkpoint selection or banking.

B.3 CIFAR-10 / CIFAR-100

Model. We use a single 2-vode conv-DEQ block, with an inner vode u and a main vode s. GroupNorm is applied only to the main vode; normalizing every prediction would make the spectral cap inert. There is no residual connection. A feedforward stem (conv–GN–ReLU $\times 2$, then a linear conv, strides $1, 2, 2$) embeds the image to 8×8 before the recurrent block; the read-out is an affine GroupNorm followed by global average pooling and a linear layer. Both datasets use a channel width of 256 (≈1.58M parameters).

Equilibrium and optimizer. Both datasets use a one-sided estimator with a coin-flip nudge sign (called *random nudging* in the literature), SiLU activations, and AdamW at lr $\approx 1.3 \times 10^{-3}$, weight decay $\approx 6 \times 10^{-5}$, batch size 256, for 120 epochs, with random crop and horizontal flip (no cutout).