The Propagation Field: A Geometric Substrate Theory of Deep Learning

Anonymous Authors[1]

Abstract

Modern deep learning treats neural networks primarily as endpoint functions from inputs to outputs. Inspired by the shift from force to geometry in physics, we ask whether a network should instead be understood through the geometry of its internal propagation. We define a neural propagation field as the collection of hidden-state trajectories and local Jacobian operators across depth. Endpoint losses constrain only the boundary behavior of this field, leaving its interior geometry underdetermined. We show that endpoint-equivalent models can differ by orders of magnitude in trajectory and Jacobian structure, and introduce observable field metrics such as path sensitivity, solver consistency, and trajectory/Jacobian retention. In controlled teacher-flow and PDE systems, endpoint fitting fails to recover the underlying propagation law. In real multi-path tasks, field-aware objectives improve unseen-path generalization, OOD robustness, and calibration when aligned with the observation structure, but can collapse when over-constrained. In continual learning, field-preservation regularization complements replay and distillation: on Split CIFAR-100, DER++ with field preservation improves average accuracy, backward transfer, and field-retention metrics. These results identify propagation-field quality as a measurable and trainable property of neural networks beyond endpoint performance.

1. Introduction

Deep learning succeeds by optimizing input–output maps through task loss, yet how information propagates internally remains poorly understood (LeCun et al., 2015). Standard training treats hidden layers as intermediate computations driven by output error (Rumelhart et al., 1986), constraining endpoint behavior but not the geometry of layerwise representations. This gap appears across adversarial robustness (Szegedy et al., 2013; Goodfellow et al., 2014), in-context learning (Brown et al., 2020; Olsson et al., 2022), continual forgetting (Kirkpatrick et al., 2017), and endpoint-loss scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), suggesting that model behavior depends on internal propagation structure as well as endpoint functions.

Inspired by classical field theory, general relativity, and Yang–Mills gauge theory, we propose a propagation-field perspective for understanding the internal mechanisms of deep learning. In physics, a central conceptual transition from Newtonian mechanics to general relativity and non-Abelian gauge theory is to reinterpret apparent "forces" as manifestations of more fundamental fields and geometric structures: gravity can be described as spacetime curvature, while fundamental interactions can be formulated through the curvature of gauge fields (Albert et al., 1916; Yang & Mills, 1954; Nakahara, 2018). We view layerwise Jacobians, hidden-state trajectories, and their geometric properties path sensitivity, curvature, and velocity alignment as defining an internal propagation field that underlies the endpoint function. The endpoint function is then a terminal projection of this field, not the entirety of what learning produces.

2. Endpoint Supervision Does Not Identify the Propagation Field

Standard supervised learning optimizes a task loss over a data distribution $\mathcal{D}$ (LeCun et al., 2015):

$$\mathcal{L}_{\text{task}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(f_\theta(x), y)\right]. \tag{1}$$

This constrains input–output behavior but does not uniquely determine internal propagation.

Definition 2.1 (Endpoint Equivalence)**.** Given a task distribution $\mathcal{D}$ and tolerance $\epsilon > 0$, two models f_{θ_1} and f_{θ_2} are *endpoint-equivalent* if

$$|\mathcal{L}_{\text{task}}(\theta_1) - \mathcal{L}_{\text{task}}(\theta_2)| \le \epsilon. \tag{2}$$

Endpoint-equivalent models need not propagate information the same way. We denote the propagation field of a deep model by

$$\Phi_\theta(x) = \{h_0, h_1, \ldots, h_L;\ J_0, J_1, \ldots, J_{L-1}\}, \tag{3}$$

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. Endpoint equivalence does not imply field equivalence. Endpoint metrics remain similar, while field metrics can differ substantially. Lower field values are better.

Setting	Endpoint	Field	Gap
Teacher A, Traj.	$0.9975/0.9975$	$3.52 \to 0.06$	$58.7\times$
Teacher A, Deriv.	$0.9975/0.9975$	$2.76 \to 0.15$	$18.4\times$
Teacher B, Traj.	$\approx .99/\approx .99$	$3.33 \to 0.07$	$48.0\times$
PDE-A, $T{=}2.0$	endpoint fit	$1.217 \to 0.195$	$6.2\times$
Tiny-ImageNet	$.097 \to .118$	$4.49 \to 1.27$	$3.5\times$

where h_ℓ is the hidden state at layer ℓ and

$$J_\ell = \frac{\partial h_{\ell+1}}{\partial h_\ell} \tag{4}$$

is the local propagation Jacobian. For continuous-flow models, Φ_θ may also include the trajectory $h(t)$, the vector field $f_\theta(h,t)$, and its numerical integration behavior.

Definition 2.2 (Propagation-Field Distance)**.** Let Φ_{θ_1} and Φ_{θ_2} be the propagation fields of two models f_{θ_1} and f_{θ_2}. A field distance is

$$\begin{aligned} d_{\text{field}}(\theta_1,\theta_2) = \mathbb{E}_{x\sim\mathcal{D}}\big[&d_{\text{traj}}(\mathcal{H}_{\theta_1}(x),\mathcal{H}_{\theta_2}(x)) \\ &+ \lambda_J d_{\text{Jac}}(\mathcal{J}_{\theta_1}(x),\mathcal{J}_{\theta_2}(x))\big] \end{aligned} \tag{5}$$

where $\mathcal{H}_\theta(x)$ denotes the hidden-state trajectory and $\mathcal{J}_\theta(x)$ denotes the collection of local Jacobians or their spectral proxies.

Proposition 2.3 (Endpoint Equivalence Does Not Imply Field Equivalence)**.** *A small discrepancy in task loss does not imply a small discrepancy in the propagation field:*

$$\mathcal{L}_{\text{task}}(\theta_1) \approx \mathcal{L}_{\text{task}}(\theta_2) \not\Rightarrow d_{\text{field}}(\theta_1,\theta_2) \approx 0. \tag{6}$$

Scaling laws therefore characterize the scaling of endpoint loss, but not necessarily the scaling of internal propagation quality (Kaplan et al., 2020; Hoffmann et al., 2022).

2.1. Experimental Evidence

Since standard benchmarks rarely measure internal propagation, we provide compact empirical evidence in Table 1. Experiment details are in Appendices A.5, A.2.4, and A.6. The teacher-flow setting gives the cleanest test: models with nearly identical endpoint accuracy can exhibit orders-of-magnitude differences in trajectory and derivative recovery. PDE extrapolation shows that endpoint fitting does not recover propagation laws, while reveal-path tasks indicate that path sensitivity is a distinct internal axis rather than a proxy for accuracy. These results support the claim that endpoint supervision identifies a behavioral equivalence class but does not uniquely determine the internal propagation field.

3. Propagation-Field Theory

The previous section showed that endpoint-equivalent models can have different hidden trajectories, Jacobian structures, and path sensitivities. We now define the measurable quantities that characterize these differences. Rather than assuming Riemannian structure or geodesic dynamics, we work directly with quantities computable in standard networks: hidden-state trajectories and local Jacobian operators.

Definition 3.1 (Observable Propagation Path)**.** Given an input x and model f_θ, let

$$h_0^\theta(x), h_1^\theta(x), \ldots, h_L^\theta(x) \tag{7}$$

be the hidden states across depth, where $h_0^\theta(x)$ is the input encoding and $h_L^\theta(x)$ is the representation before the prediction head. Since hidden dimensions may vary across layers, we introduce a layer embedding map $\phi_\ell : \mathcal{H}_\ell \to \mathbb{R}^d$ and define the standardized hidden state

$$z_\ell^\theta(x) = \phi_\ell(h_\ell^\theta(x)). \tag{8}$$

If all layers share the same dimension, ϕ_ℓ is the identity. The **observable propagation path** is

$$\gamma_\theta(x) = \{z_0^\theta(x), z_1^\theta(x), \ldots, z_L^\theta(x)\}. \tag{9}$$

Definition 3.2 (Local Propagation Operator)**.** At layer ℓ, the local propagation operator is

$$J_\ell^\theta(x) = \frac{\partial h_{\ell+1}^\theta(x)}{\partial h_\ell^\theta(x)}. \tag{10}$$

For a perturbation v at layer ℓ, the first-order change at layer $\ell{+}1$ is $\delta h_{\ell+1} \approx J_\ell^\theta(x)v$. Thus $J_\ell^\theta(x)$ describes how a single layer amplifies, compresses, or redirects perturbations in hidden space.

Definition 3.3 (Jacobian-Induced Propagation Metric)**.** The local Jacobian induces a layerwise metric

$$G_\ell^\theta(x) = (J_\ell^\theta(x))^\top J_\ell^\theta(x), \tag{11}$$

for which $v^\top G_\ell^\theta(x) v = \|J_\ell^\theta(x)v\|_2^2$. Strong amplification indicates sensitivity; strong compression indicates information loss. In high-dimensional networks, $G_\ell^\theta(x)$ is not formed explicitly; experiments use scalable proxies such as Jacobian-vector products, random projections, and spectral estimates.

3.1. Accumulated Propagation Metric and Global Perturbation Transport

Definition 3.4 (Accumulated Propagation Operator)**.** The accumulated Jacobian from layer 0 to layer ℓ is

$$\bar{J}_{0:\ell}^\theta(x) = J_{\ell-1}^\theta(x) \cdots J_1^\theta(x) J_0^\theta(x), \tag{12}$$

with the corresponding accumulated metric $\bar{G}_{0:\ell}^\theta(x) = (\bar{J}_{0:\ell}^\theta(x))^\top \bar{J}_{0:\ell}^\theta(x)$.

Lemma 3.5 (First-Order Perturbation Propagation)**.** *For a small input perturbation δh_0, the displacement at depth ℓ satisfies*

$$h_\ell^\theta(x+\delta x) - h_\ell^\theta(x) \approx \bar{J}_{0:\ell}^\theta(x)\,\delta h_0, \tag{13}$$

and therefore

$$\|h_\ell^\theta(x+\delta x) - h_\ell^\theta(x)\|_2^2 \approx \delta h_0^\top \bar{G}_{0:\ell}^\theta(x)\delta h_0. \tag{14}$$

The spectral structure of $\bar{G}_{0:\ell}^\theta(x)$ therefore governs perturbation sensitivity: a large eigenvalue along a given direction means small input perturbations in that direction can be amplified through depth. This is a first-order approximation; for large perturbations it serves as a geometric explanation rather than an exact identity.

To make propagation fields measurable, we use three discrete geometry metrics.

Path length.

$$\mathrm{Len}_\theta(x) = \sum_{\ell=0}^{L-1} \left\| z_{\ell+1}^\theta(x) - z_\ell^\theta(x) \right\|_2. \tag{15}$$

This measures the total representation displacement across depth.

Discrete curvature. Let

$$\Delta z_\ell^\theta(x) = z_{\ell+1}^\theta(x) - z_\ell^\theta(x). \tag{16}$$

We define

$$\kappa_\theta(x) = \frac{1}{L-1}\sum_{\ell=1}^{L-1} \frac{\left\| z_{\ell+1}^\theta(x) - 2z_\ell^\theta(x) + z_{\ell-1}^\theta(x) \right\|_2}{\left\| z_{\ell+1}^\theta(x) - z_\ell^\theta(x) \right\|_2^2 + \varepsilon}. \tag{17}$$

This is a discrete proxy for the bending of the hidden trajectory, not a strict Riemannian curvature.

Velocity alignment.

$$\mathrm{Align}_\theta(x) = \frac{1}{L-1}\sum_{\ell=1}^{L-1} \frac{\left\langle \Delta z_\ell^\theta(x), \Delta z_{\ell-1}^\theta(x) \right\rangle}{\left\| \Delta z_\ell^\theta(x) \right\|_2 \left\| \Delta z_{\ell-1}^\theta(x) \right\|_2 + \varepsilon}. \tag{18}$$

High alignment indicates consistent propagation direction; low or negative alignment indicates oscillation or reversal. These metrics correspond to Path, Curvature, and Velocity Align in our experiments.

Many tasks admit multiple equivalent observation paths, such as image patch orders, frequency decompositions, temporal speech windows, or different textual evidence orders. For two paths p, q of the same sample x, let

$$h_L^{\theta,p}(x), \quad h_L^{\theta,q}(x), \quad o^{\theta,p}(x), \quad o^{\theta,q}(x) \tag{19}$$

denote the final hidden states and logits.

We define hidden-state path sensitivity as

$$\mathrm{PathSens}_h = \mathbb{E}_{x,p,q}\left[\left\| h_L^{\theta,p}(x) - h_L^{\theta,q}(x) \right\|_2 \right], \tag{20}$$

and output-level path sensitivity as

$$\mathrm{PathSens}_o = \mathbb{E}_{x,p,q}\left[\left\| o^{\theta,p}(x) - o^{\theta,q}(x) \right\|_2 \right]. \tag{21}$$

Lower path sensitivity indicates greater invariance to observation paths, but it does not guarantee better task performance: collapsed representations can also yield low sensitivity. We therefore evaluate it together with accuracy, OOD generalization, calibration, and representation variance.

For shared-field or continuous-depth models, the same underlying field can be evaluated under different discretizations. Let

$$h_T^{(s)}(x), \quad o_T^{(s)}(x) \tag{22}$$

be the final hidden state and logits under solver or step configuration s. We define

$$\mathrm{SolverErr} = \mathbb{E}_{x,s_1,s_2}\left[\left\| h_T^{(s_1)}(x) - h_T^{(s_2)}(x) \right\|_2 + \left\| o_T^{(s_1)}(x) - o_T^{(s_2)}(x) \right\|_2 \right]. \tag{23}$$

Low solver error, especially under step refinement, suggests that layers behave as numerical slices of a shared propagation field rather than as a fixed-depth black box.

In continual learning, new tasks may overwrite not only the output function but also the internal propagation field of old tasks. Let θ^{old} and θ^{new} denote parameters before and after learning a new task, and let $\mathcal{A}$ be old-task anchor samples.

Definition 3.6 (Trajectory Retention Score)**.**

$$\mathrm{FRS} = \mathbb{E}_{x\in\mathcal{A}} \frac{1}{L+1}\sum_{\ell=0}^{L} \left\| h_\ell^{\theta^{\mathrm{old}}}(x) - h_\ell^{\theta^{\mathrm{new}}}(x) \right\|_2^2. \tag{24}$$

Lower FRS indicates better preservation of old-task hidden trajectories.

Definition 3.7 (Jacobian Retention Score)**.**

$$\mathrm{JRS} = \mathbb{E}_{x\in\mathcal{A},\ell,\delta}\left[\frac{\left\| J_\ell^{\theta^{\mathrm{old}}}(x)\delta - J_\ell^{\theta^{\mathrm{new}}}(x)\delta \right\|_2^2}{\|\delta\|_2^2 + \varepsilon} \right]. \tag{25}$$

Lower JRS indicates better preservation of old-task local propagation operators.

FRS measures hidden-trajectory drift, while JRS measures Jacobian-field drift. They capture field-level forgetting, complementing standard function-level continual learning metrics such as AA and BWT.

3.2. Field-aware Optimization Objectives

Propagation geometry can also be optimized. We define multi-path consistency as

$$\mathcal{L}_{\text{reveal}} = \mathbb{E}_{x,p,q}\Big[\Big\|h_L^{\theta,p}(x) - h_L^{\theta,q}(x)\Big\|_2^2 + \mathrm{KL}\big(\sigma(o^{\theta,p}(x)) \,\|\, \sigma(o^{\theta,q}(x))\big)\Big]. \tag{26}$$

Solver consistency is

$$\mathcal{L}_{\text{solver}} = \mathbb{E}_{x,s_1,s_2}\Big[\Big\|h_T^{(s_1)}(x) - h_T^{(s_2)}(x)\Big\|_2^2 + \Big\|o_T^{(s_1)}(x) - o_T^{(s_2)}(x)\Big\|_2^2\Big]. \tag{27}$$

Jacobian-field smoothness is

$$\mathcal{L}_{\text{jac}} = \mathbb{E}_{x,\ell,\delta}\left[\frac{\|J_\ell^\theta(x)\delta - J_{\ell-1}^\theta(x)\delta\|_2^2}{\|\delta\|_2^2 + \varepsilon}\right]. \tag{28}$$

The general field-aware loss is

$$\mathcal{L}_{\text{field}} = \lambda_r \mathcal{L}_{\text{reveal}} + \lambda_s \mathcal{L}_{\text{solver}} + \lambda_J \mathcal{L}_{\text{jac}}, \tag{29}$$

and the training objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{field}}. \tag{30}$$

For continual learning, we combine field preservation with replay or distillation:

$$\mathcal{L}_{\text{CL}} = \mathcal{L}_{\text{new}} + \mathcal{L}_{\text{replay/distill}} + \lambda_{\text{FPR}}\mathcal{L}_{\text{FPR}}, \tag{31}$$

where

$$\mathcal{L}_{\text{FPR}} = \lambda_h \mathbb{E}_{x,\ell}\left[\Big\|h_\ell^{\theta^{\text{old}}}(x) - h_\ell^\theta(x)\Big\|_2^2\right] + \lambda_J \mathbb{E}_{x,\ell,\delta}\left[\frac{\Big\|J_\ell^{\theta^{\text{old}}}(x)\delta - J_\ell^\theta(x)\delta\Big\|_2^2}{\|\delta\|_2^2 + \varepsilon}\right]. \tag{32}$$

FPR is not a replacement for replay or distillation; it adds the missing propagation-structure constraint. Conventional methods preserve old-task discriminative functions, while FPR preserves their internal propagation geometry.

3.3. Field Collapse and Discriminative Constraints

Field-aware regularization can be harmful when it is too strong. A model exhibits field collapse when field-consistency metrics improve while discriminative performance degrades. Typical indicators include

$$\mathrm{Var}_x\left[h_L^\theta(x)\right] \to 0, \tag{33}$$

or prediction homogenization:

$$p_\theta(y \mid x) \approx p_0(y), \quad \forall x. \tag{34}$$

Thus,

$$\text{PathSens} \downarrow \not\Rightarrow \text{Accuracy} \uparrow. \tag{35}$$

Effective field-aware learning must therefore combine propagation consistency with discriminative and anti-collapse constraints.

3.4. Experiment

We evaluate both endpoint behavior and internal propagation geometry, organized around four questions: **Q1** whether endpoint supervision uniquely identifies the propagation field; **Q2** whether field-aware objectives improve multi-path, OOD, or calibration performance; **Q3** whether field consistency can over-regularize and cause collapse; and **Q4** whether field preservation improves continual learning when combined with replay or distillation. Experiment details are in Appendix A.

Main Analysis Table 2 summarizes six representative results, ordered from controlled systems to real tasks and continual learning. In teacher-flow, two models achieve identical accuracy (0.9975) but differ by $59\times$ in trajectory recovery: field supervision reduces trajectory error from 3.52 to 0.06 without changing endpoint performance. In PDE extrapolation, endpoint maps fit the training horizon but fail at $T = 2.0$ $(\mathrm{MSE} = 1.217)$, whereas generator-style propagation models reach $\mathrm{MSE} = 0.195$ with energy correlation 0.995. These results show that endpoint loss identifies a behavioral equivalence class, not the underlying propagation law. On real multi-path tasks, field-aware objectives help only when aligned with task structure. On Tiny-ImageNet, the full objective improves accuracy $(0.097 \to 0.118)$, unseen-path accuracy, OOD accuracy, and calibration while reducing path sensitivity. On CIFAR-100, however, it reduces path sensitivity but collapses accuracy to 0.0093, showing that field consistency alone is not sufficient. Continual learning shows the same complementarity: standalone FPR preserves field metrics $(\mathrm{FRS} = 0.557, \mathrm{JRS} = 0.033)$ but not endpoint performance $(\mathrm{AA} = 0.037)$, while DER++ preserves endpoint performance $(\mathrm{AA} = 0.135)$ but not field geometry $(\mathrm{JRS} = 0.177)$. Combining them recovers both: DER++ + FPR-Full at 200 samples/task improves AA $(0.125 \to 0.136)$, BWT $(-0.538 \to -0.481)$, and JRS $(0.166 \to 0.091)$, indicating that field retention and function retention are complementary objectives.

Controlled Evidence: Endpoint Behavior Does Not Identify the Field Table 13 summarizes the controlled evidence. In teacher-flow systems, endpoint accuracy remains essentially unchanged after adding field supervision, yet trajectory and derivative errors drop by one to two orders of magnitude. This directly supports the underdetermination claim: endpoint-equivalent models need not be field-equivalent. In PDE systems, endpoint maps achieve very

Table 2. **Compact experimental evidence for propagation-field geometry.** We report representative results across controlled propagation, real multi-path transfer, collapse cases, and continual learning. Task metrics measure endpoint behavior or function retention; field metrics measure hidden trajectory, local Jacobian, or path consistency. Lower field metrics are better.

Claim	Setting	Comparison	Task / Endpoint Metric	Generalization / Calibration	Propagation-Field Metric
Endpoint does not identify field	Teacher-flow A	FlowEndOnly → FlowFieldLoss	Acc .9975 → .9975	Reparam. final 2.45×10^{-3} → 1.23×10^{-3}	Traj. 3.52 → .059; Deriv. 2.76 → .150
Endpoint fitting does not recover law	PDE-A extrapolation	Endpoint map → generator-style field	Trained endpoint fit	T=2 MSE 1.217 → .195	Energy corr. N/A → .995; regrid err. 2.17×10^{-2}
Field helps when task-aligned	Tiny-ImageNet reveal	Task → Full field-aware	Acc .097 → .118	Unseen .085 → .109; OOD .100 → .128; ECE .102 → .018	PathSens-logit 4.49 → 1.27; hidden 4.44 → 2.07
Over-constraint can collapse	CIFAR-100 reveal	Task → Full field-aware	Acc .0405 → .0093	Unseen .0295 → .0090; OOD .057 → .010	PathSens-logit 15.19 → 3.04; hidden 7.28 → 2.14
Field retention differs from function retention	Split CIFAR-100 CL	DER++ vs. FPR-Traj	AA .1346 vs. .0374	BWT strong for replay; weak for standalone FPR	FRS .9209 → .5568; JRS .1768 → .0332
FPR complements replay/distillation	Split CIFAR-100 CL, budget 200	DER++ → DER++ + FPR-Full	AA .125 → .136; BWT −.538 → −.481	Hybrid improves function retention	FRS .980 → .714; JRS .166 → .091

low training-horizon error but fail at $T = 2.0$, whereas propagation-style models substantially reduce extrapolation error and preserve energy evolution. Thus, fitting endpoints is not equivalent to learning the underlying propagation law.

Real Multi-path Transfer: When Field Objectives Help We test whether field-aware objectives improve performance on real multi-path tasks. Table 15 shows that gains are task-dependent. Tiny-ImageNet provides the clearest positive result: the full objective improves AccAvg, UnseenAcc, OOD accuracy, and ECE while reducing both logit and hidden path sensitivity. CIFAR-100 shows a weaker signal: reveal consistency improves accuracy and lowers PathSens, but the full objective degrades accuracy despite further reducing path sensitivity. AG News and SST-2 show no consistent improvement, with the full objective reducing AG News accuracy to near-random. Speech Commands yields modest OOD gains but limited ID improvement. Overall, field-aware objectives help when path consistency aligns with task structure, but can collapse discriminative performance when imposed without task-aligned constraints.

Field Collapse: Low Path Sensitivity Is Not Sufficient As shown in Table 15, lower path sensitivity does not necessarily imply better task performance. For example, the full objective on CIFAR-100 reduces PathSensLogit from 15.1867 to 3.0355, but AccAvg drops from 0.0405 to 0.0093. Similarly, AG News already has low PathSens under task-only training, while reveal/full objectives degrade accuracy.

Field-aware Objectives and Optimization Tradeoffs Table 9 summarizes objective-level effects, and Table 16 summarizes training-level Pareto behavior. Reveal consistency is the strongest single component, increasing AccAvg from 0.3472 to 0.8461 and reducing PathSensLogit from 25.6669 to 6.9140. The full objective yields the lowest PathSensHidden (3.9182) and JacWDist (0.0032), but does not exceed reveal-only accuracy. At the optimization level, different algorithms occupy different Pareto points: LocalFieldMatch improves over FullBPTT in both TestAcc (0.615 vs. 0.590) and TrajRMSE (1.034 vs. 1.053), while gradient-cosine analysis shows task–field conflict with negative-gradient fractions up to 0.36. Standard end-to-end backpropagation does not automatically find the best task–field trade-off.

Continual Learning: Field Retention as a Complementary Objective Table 17 shows that field retention complements function retention in Split CIFAR-100. Trajectory drift is weakly but significantly correlated with forgetting ($r = 0.1748$, $p = 0.0158$; $\rho = 0.1498$, $p = 0.0391$), while Jacobian and parameter drift are not significant predictors. Replay and distillation methods are stronger for endpoint retention: DER++ achieves AA 0.1346 and BWT −0.5248. Standalone FPR mainly improves field metrics—FPR-Traj obtains the lowest JRS (0.0332). The strongest result is the hybrid setting: DER++ + FPR-Full at 200 samples/task improves AA, BWT, FRS, and JRS simultaneously (+0.0111, +0.0571, −0.2661, −0.0748). FPR is not a replacement for replay, but a field-retention regularizer that complements existing methods.

4. Conclusion

We argue that a deep network is an endpoint function at its boundary, but a propagation field in its interior. We formalize this field through hidden trajectories and local Jacobian operators, show that endpoint supervision underdetermines it, and introduce metrics and objectives to measure, regularize, and preserve it. Across controlled, multi-path, and continual learning settings, propagation-field quality emerges as a distinct, trainable property beyond endpoint performance. Finally, our collapse results suggest a Kakeya-like directional principle: useful propagation fields must reduce spurious path sensitivity while preserving sufficient directional richness for discrimination (Wang & Zahl, 2026).

References

Albert, E., Perrett, W., and Jeffery, G. The foundation of the general theory of relativity. *Annalen der Physik*, 354(7): 769, 1916.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Nakahara, M. *Geometry, topology and physics*. CRC press, 2018.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Wang, H. and Zahl, J. Sticky kakeya sets and the sticky kakeya conjecture. *Journal of the American Mathematical Society*, 39(2):515–585, 2026.

Yang, C. N. and Mills, R. L. Conservation of isotopic spin and isotopic gauge invariance. *Phys. Rev.*, 96:191–195, Oct 1954. doi: 10.1103/PhysRev.96.191. URL https://link.aps.org/doi/10.1103/PhysRev.96.191.

Table 3. Unified experimental objects used across the study.

Object	Definition	Role in Experiments
Endpoint Map	Directly learns a mapping from inputs to labels or endpoint states.	Serves as the baseline corresponding to the endpoint-function view.
Time-Conditioned Discrete Flow	Learns a family of discrete propagation maps conditioned on time or layer index.	Tests whether adding temporal or depth conditioning improves internal propagation structure.
Shared-Field Discrete Flow	Shares the same propagation parameters across layers and advances states through Euler-like updates.	Tests whether layers can be interpreted as discrete slices of a common propagation field.
Continuous Flow	Explicitly learns a continuous vector field and obtains terminal states through numerical integration.	Evaluates solver and step-size reparameterization consistency.
Residual Stack	Uses a standard residual network with independent parameters at each layer.	Serves as a conventional deep discrete-architecture baseline.
Hybrid Flow	Combines a shared propagation field with a small number of learnable correction terms.	Balances continuous-field constraints with expressive flexibility.

A. Experimental Overview and Unified Protocol

This series of experiments is designed around a central question: whether the internal computation of deep learning models can be understood as a *propagation field* evolving along depth, time, or observation paths, and whether such a field can serve as a measurable, trainable, and retainable object in machine learning. Unlike the conventional endpoint-function view, which primarily evaluates the mapping from inputs to outputs, our experimental framework explicitly examines hidden trajectories, local Jacobians, path consistency, solver consistency, and cross-task retention of internal propagation structures.

A.1. Core Hypotheses

We organize the experiments around the following hypotheses:

- **Endpoint supervision is underdetermined.** Standard task losses typically constrain only final labels or endpoint outputs. As a result, they may identify a behavioral equivalence class rather than a unique internal propagation field.
- **Propagation quality can become useful when aligned with task structure.** When a task involves path dependence, sequential dependence, partial observability, multi-view evidence, or long-horizon process structure, the quality of the internal propagation field may translate into measurable learning benefits.
- **Propagation quality is multi-dimensional.** It cannot be reduced to a single scalar measure, but is jointly characterized by state trajectories, local propagation operators, path consistency, and numerical reparameterization consistency.
- **Forgetting may occur at the field level.** In continual learning, forgetting may not only appear as degradation of endpoint performance, but also as rewriting of the internal propagation field associated with previous tasks. However, field retention and accuracy retention are not necessarily equivalent.

A.1.1. Unified Experimental Objects

Across experiments, we compare several model families that instantiate different assumptions about endpoint mappings and internal propagation structures. Table 3 summarizes their roles.

A.1.2. Unified Evaluation Metrics

We evaluate models along two complementary axes: external task performance and internal field quality. The former measures whether the model produces correct endpoint behavior, while the latter measures how information is propagated

Table 4. Unified evaluation metrics for endpoint performance and internal propagation quality.

Category	Metric	Meaning
External Task Metrics	ID Accuracy	Classification or prediction accuracy on the in-distribution test set.
External Task Metrics	OOD Accuracy	Accuracy under distributional perturbations, such as noise, path shifts, center shifts, rotations, or truncations.
External Task Metrics	Unseen-Path Accuracy	Accuracy on reveal paths or observation paths not encountered during training.
External Task Metrics	Low-Shot Accuracy	Task performance under few-shot training conditions.
External Task Metrics	ECE	Expected Calibration Error, used to evaluate confidence calibration.
Internal Field Metrics	PathSensLogit / PathSensHidden	Differences in output logits or hidden states for the same sample under different reveal paths.
Internal Field Metrics	Trajectory RMSE	Root mean squared error between the predicted trajectory and the teacher-provided ground-truth trajectory.
Internal Field Metrics	Derivative RMSE	Error between the learned local vector field and the ground-truth vector field.
Internal Field Metrics	Solver Consistency	Consistency of terminal states or logits when the same model is evaluated with different integration step sizes or solvers.
Internal Field Metrics	Jacobian Smoothness / JRS	Difference between local Jacobians across adjacent layers, or between old and new models in continual learning.
Internal Field Metrics	NormPath / Curvature / VelAlign	Measures of hidden trajectory length, bending, and alignment of adjacent velocity directions.

internally. Table 4 summarizes the metrics used throughout the experiments.

A.2. PDE Prototype Experiments: From Endpoint Functions to Propagation Generators

The PDE prototype experiments are designed to construct controlled systems with known ground-truth propagation laws, and to compare models that learn only endpoint mappings with models that learn propagation generators. The goal is not to optimize performance on realistic vision tasks, but to turn the question of whether a propagation field is learnable into a controlled numerical experiment. Specifically, we evaluate models in terms of horizon extrapolation, semigroup consistency, step-size reparameterization, and perturbation propagation.

A.2.1. Data and Dynamical Systems

The input is an initial one-dimensional or low-dimensional field state $u(x, 0)$, generated from combinations of smooth basis functions, localized wave packets, or random initial conditions. A known numerical PDE solver is then used to generate the full trajectory $u(x, t)$, from which terminal states are sampled at multiple horizons.

We consider three PDE families:

- **PDE-A: Linear advection–diffusion.** This system is used to evaluate smooth propagation, diffusion, and temporal extrapolation.

Table 5. Model configurations in the PDE prototype experiments.

Model	Training Objective	Experimental Role
M1: Fixed-Endpoint MLP	Directly predicts the terminal state at the training horizon from the initial state.	Tests whether memorizing a fixed endpoint map can extrapolate to other time horizons.
M2: Time-Conditioned MLP	Takes the initial state and target time T as inputs and directly predicts $u(T)$.	Tests whether adding time conditioning is sufficient to learn a family of propagation maps.
M3: Propagation Generator	Learns a local update rule or vector field and obtains the terminal state through multi-step integration.	Tests whether the model learns a reparameterizable propagation mechanism rather than a direct endpoint map.
M4: Structured Propagation Model	Adds PDE-style inductive structure or stronger propagation constraints on top of the learned generator.	Tests whether explicit propagation structure improves extrapolation and stability.

- **PDE-B: Nonlinear advection–diffusion–reaction.** This system introduces nonlinear reaction terms, source effects, and local perturbation propagation.
- **PDE-C: Non-autonomous variable-coefficient dynamics.** In this system, coefficients vary over time or space, allowing us to test whether models can learn non-stationary propagation laws.

A.2.2. Model Configurations

Table 5 summarizes the model families used in the PDE prototype experiments.

A.2.3. Training and Evaluation Protocol

The training and evaluation procedure is as follows:

1. Generate training initial conditions, test initial conditions, and ground-truth PDE trajectories across multiple horizons.
2. Train endpoint models, time-conditioned models, and propagation-generator models on the training horizons.
3. Evaluate terminal-state prediction at multiple horizons, such as $T = 0.5, 1.0, 1.5,$ and 2.0.
4. For generator-based models, change the numerical integration step size, such as halving dt, and test whether the model preserves the same propagated state.
5. Add small perturbations to the initial state and compare the model-induced perturbation propagation with the ground-truth PDE perturbation propagation.
6. Include negative controls, such as random state pairing, shuffled time order, or shuffled trajectories, to test whether the model relies on the true temporal propagation order.

A.2.4. Metrics and Outputs

We use the following metrics to evaluate whether a model learns only endpoint behavior or a stable propagation law:

- **Endpoint MSE:** mean squared error between the predicted terminal state and the ground-truth terminal state.
- **Horizon extrapolation MSE:** terminal-state error evaluated outside the training time horizon.
- **Semigroup consistency:** consistency between directly integrating to T and composing two shorter integrations, e.g., $T_1 + T_2 = T$.

- **Regrid $dt/2$ error:** change in model output when the integration step size is halved.
- **Energy correlation:** correlation between the energy evolution of the model trajectory and that of the ground-truth PDE trajectory.
- **Perturbation MSE:** error between the ground-truth and model-predicted propagation of small initial perturbations.
- **Negative-control gap:** performance gap between the correctly ordered trajectories and randomized or shuffled trajectory controls.

A.3. Propagation-Field Extraction in Ordinary Deep Networks

This group of experiments investigates whether the intermediate representations of standard deep architectures, such as ResNets and Transformers, can also be analyzed as trajectories or fields evolving along depth, even in the absence of an explicit PDE teacher. To this end, we treat the layer index as a pseudo-time variable, and record intermediate hidden states, local Jacobian spectra, trajectory curvature, and reparameterization behavior across depth.

A.3.1. ResNet Continuization and Trajectory Alignment

This experiment examines whether the hidden representations of ResNets exhibit trajectory-like structure when depth is interpreted as pseudo-time.

- We train ResNets with different depths, e.g., depth $= 4, 6, 12$.
- For the same test samples, we extract the hidden representation after each residual block.
- For each sample, the sequence of layerwise hidden states is treated as a trajectory along the depth direction.
- We compare the resulting trajectory shapes across networks of different depths using alignment procedures such as PCA-based projection or Procrustes alignment.
- We record **NormPath**, **Curvature**, **VelAlign**, and the **pairwise alignment error across depths**.

The purpose of this experiment is to test whether ResNet trajectories admit a continuous-depth interpretation, rather than behaving as unrelated layerwise transformations.

A.3.2. Comparing Depth Fields in Transformers and ResNets

This experiment compares the propagation statistics of Transformers and ResNets to determine whether propagation-field phenomena are specific to residual networks or arise more broadly in deep architectures.

- We construct Transformers with genuine multi-token inputs rather than degenerate single-token attention.
- Intermediate outputs from Transformer blocks and ResNet blocks are mapped into a comparable hidden space.
- We compare whether the two architectures exhibit similar depthwise propagation statistics, such as high **VelAlign** or low **Curvature**.
- The goal is to distinguish whether the propagation-field phenomenon is a special case of residual architecture design or a more general property of deep representation evolution.

A.3.3. Jacobian Spectral Smoothness Analysis

This experiment studies whether layerwise transformations evolve smoothly across depth, or instead change abruptly from layer to layer.

- For each layer or block, we estimate the local Jacobian spectrum using randomized projections, e.g., Jacobian–vector products (Jv) or finite-difference approximations.

Table 6. Model families compared in the layerwise propagation experiment.

Model Family	Definition	Experimental Role
Endpoint Map	Directly learns a mapping from input to label, without explicitly constraining internal propagation.	Serves as the endpoint-only baseline.
Time-Conditioned Family	Takes time or layer index as an input and learns a family of ordered mappings.	Tests whether temporal/depth conditioning alone induces structured propagation.
Continuous Flow	Explicitly learns a continuous vector field and obtains outputs through numerical integration.	Represents the strongest propagation-field formulation.

- We compute the Wasserstein distance between the Jacobian spectral distributions of adjacent layers.
- We also compute the entropy of the spectral distribution at each layer to examine whether the local propagation operator changes smoothly along depth.
- This experiment is designed to test whether inter-layer transformations resemble discrete jumps or the evolution of local operators in a continuous field.

A.3.4. Depth Refinement and Solver Reparameterization

This experiment directly evaluates whether layers can be interpreted as numerical slices of a common propagation field.

- We fix a learned field in a **Continuous Flow** or **Shared-Field** model.
- We evaluate multiple discretization depths, e.g., $8, 16, 32, 64$, and 128 steps.
- We also evaluate different numerical solvers, such as Euler, midpoint, and RK4.
- Across these discretizations, we compare the terminal hidden-state error, logit error, trajectory error, and prediction agreement.

If the model truly represents a coherent propagation field, its predictions should remain stable under refinement of the discretization or changes in the solver.

A.3.5. Layerwise Comparison of Endpoint Maps, Time-Conditioned Families, and Continuous Flows

This experiment compares three model families under the same task, with the goal of separating *functional equivalence* from *equivalence in internal propagation geometry*. Table 6 summarizes the comparison.

For these three families, we compare:

- hidden trajectory length,
- curvature,
- alignment of velocity directions across depth, and
- task accuracy.

The purpose of this comparison is to distinguish whether two models that are functionally similar at the endpoint level also exhibit similar internal propagation geometry.

A.4. Hard-Manifold Classification and True Reparameterization Experiments

In the initial experiments, many models achieved high in-distribution accuracy, making it difficult to determine whether propagation-field constraints were genuinely useful. To avoid this saturation effect, we further design harder synthetic manifold classification benchmarks and include OOD evaluation, low-shot regimes, perturbation tests, and true depth-reparameterization experiments.

A.4.1. Hard-Manifold Data Generation

We construct multiple low-dimensional curved manifolds embedded in a high-dimensional ambient space. Each class corresponds to a manifold segment characterized by a class center, subspace directions, and nonlinear deformation terms. Task difficulty is controlled by the distance between class centers, noise magnitude, subspace overlap, and manifold curvature.

We consider three difficulty levels: **easy**, **medium**, and **hard**. These settings are designed to prevent all models from reaching saturated accuracy and to make the task sensitive to both functional expressivity and internal propagation structure.

OOD evaluation is constructed through several controlled distribution shifts:

- stronger input noise,
- rotations of the underlying subspaces,
- shifts of class centers, and
- changes in the nonlinear curvature terms.

We also evaluate low-shot learning regimes with 20, 50, and 100 samples per class.

A.4.2. Models and Fairness Controls

We compare several model families, including **Endpoint**, **TimeCond**, **SharedField**, and **Continuous** models. Whenever possible, hidden dimensions and parameter counts are controlled to make the comparison fair across architectures.

All models are trained using the same optimizer, number of training epochs, batch size, and data splits. For each difficulty level and random seed, we record both functional metrics and geometric metrics. The purpose of this experiment is not to rank models solely by accuracy, but to examine whether geometric smoothness has a conditional relationship with OOD robustness and low-shot generalization.

A.4.3. Reparameterization Experiments

To test whether a learned model can be interpreted as a discretization of a common propagation field, we first train a single learned field and then vary the number of integration steps and the numerical solver only at test time.

Importantly, we do not retrain separate networks at different depths. This avoids conflating "different models" with "different numerical slices of the same field." Instead, the same learned field is evaluated under different discretization schemes.

We record the following metrics:

- **PredAgreeRef:** prediction agreement with a reference discretization;
- **FinalHiddenErr:** error in the terminal hidden state relative to the reference solution;
- **TrajectoryErr:** discrepancy between trajectories under different discretizations;
- **LogitErr:** discrepancy between output logits under different solvers or step sizes.

If the errors decrease as the number of integration steps increases or as the solver order improves, the result supports the interpretation that the learned representation behaves like a numerically discretized propagation field.

Table 7. Task definitions in the teacher-flow experiments.

Task	Label Definition	Experimental Role
Task A: Endpoint Label	The label is determined only by the region, angle, or a linear classifier applied to the terminal state $z(T)$.	Tests whether supervision on endpoint labels is sufficient to recover the true propagation field.
Task B: Path Label	The label is determined by a path-dependent quantity, such as a trajectory integral, winding direction, or path event.	Tests whether path-dependent labels provide stronger identifiability of the propagation field.

A.5. Controlled Teacher-Flow Experiments: Endpoint Function or Propagation Field?

The teacher-flow experiments construct a ground-truth latent ODE in which both the true trajectory and the true derivative field are accessible. Their central purpose is to remove the ambiguity present in real tasks, where the true internal field is unobserved, and to directly test whether endpoint supervision is sufficient to recover the underlying propagation field.

A.5.1. Teacher Dynamics

We define a latent state $z(t)$ in a two-dimensional or low-dimensional space. The ground-truth teacher vector field is denoted by

$$\frac{dz(t)}{dt} = f^\star(z(t), t), \tag{36}$$

where $f^\star$ includes rotational, contractive, and nonlinear components. Starting from randomly sampled initial states $z(0)$, we numerically integrate the teacher dynamics to obtain the full latent trajectory

$$\{z(t_0), z(t_1), \ldots, z(t_K)\}. \tag{37}$$

The observed input x is then generated from the latent initial state through a fixed observation map,

$$x = \psi(z(0)). \tag{38}$$

This construction makes the latent trajectory and derivative field available for evaluation, while the model receives only the observed input and task labels during standard endpoint-supervised training.

A.5.2. Task Definitions

We consider two task types, summarized in Table 7.

A.5.3. Models and Losses

We compare three model classes:

- **M1: EndpointMLP.** This model directly predicts the task label from the observed input x, without explicitly modeling latent trajectories.
- **M2: FlowClassifier with endpoint-only loss.** This model uses a flow-based architecture, but is trained only with endpoint label supervision.
- **M3: FlowClassifier with field losses.** This model uses the same flow-based architecture as M2, but augments the endpoint loss with trajectory loss, derivative loss, and solver-consistency loss.

The key controlled comparison is between M2 and M3: they use the same model class, but differ only in the training objective. This allows us to isolate whether endpoint supervision alone is sufficient for field recovery.

The endpoint-only objective is

$$\mathcal{L}_{\text{endpoint}} = \mathcal{L}_{\text{task}}. \tag{39}$$

Table 8. Reveal-path construction across modalities.

Task Type	Reveal-Path Examples	Description
Image Tasks	Center, rows, spiral, frequency.	The same image is gradually presented through different spatial or frequency-domain paths.
Text Tasks	Prefix reveal, sentence-level reveal, evidence-fragment reveal.	The same text or semantic evidence is exposed in different orders.
Speech Tasks	Temporal windows, masked chunks, progressive frequency-band reveal.	The same speech command is gradually provided through different temporal or time-frequency windows.

The field-aware objective is

$$\mathcal{L}_{\text{field}} = \mathcal{L}_{\text{task}} + \alpha\mathcal{L}_{\text{traj}} + \beta\mathcal{L}_{\text{deriv}} + \gamma\mathcal{L}_{\text{solver}}, \tag{40}$$

where

$$\mathcal{L}_{\text{traj}} = \sum_{k=1}^{K} \|\hat{z}(t_k) - z(t_k)\|_2^2, \tag{41}$$

and

$$\mathcal{L}_{\text{deriv}} = \sum_{k=1}^{K} \|f_\theta(\hat{z}(t_k), t_k) - f^\star(z(t_k), t_k)\|_2^2. \tag{42}$$

The solver-consistency term $\mathcal{L}_{\text{solver}}$ measures whether the same learned field produces consistent predictions under different numerical solvers or integration step sizes.

A.5.4. Evaluation Metrics

We evaluate both endpoint task performance and internal field recovery:

- **Task Accuracy:** whether the model predicts the correct class label.
- **Trajectory RMSE:** root mean squared error between the predicted trajectory and the teacher trajectory.
- **Derivative RMSE:** error between the learned vector field and the teacher vector field.
- **Reparameterization Consistency:** output consistency of the same learned field under different solvers or integration step counts.
- **NormPath / Curvature / VelAlign / JacWDist:** geometric measures of internal field quality, including trajectory length, bending, velocity alignment, and Wasserstein distance between local Jacobian spectra.

A.6. Reveal-Path Tasks and Real/Semi-Real Task Transfer Experiments

This group of experiments transfers the propagation-field idea from controlled teacher-flow settings to scenarios closer to real machine learning tasks. The core idea is that the same input object can be gradually revealed to a model through different observation paths. If the label is invariant to the reveal path, an ideal model should form path-consistent terminal representations and predictions.

A.6.1. Reveal-Path Construction

We construct reveal paths for image, text, and speech tasks. Each reveal path defines an ordered sequence of partial observations of the same underlying input. Table 8 summarizes the path types used across modalities.

Table 9. Objective ablations for reveal-path and field-aware training.

Objective	Loss Components	Experimental Purpose
`task`	Task loss only, such as classification loss.	Serves as the standard endpoint-supervision baseline.
`reveal`	Task loss plus hidden-state or logit consistency across reveal paths.	Tests whether path consistency improves unseen-path generalization or OOD robustness.
`solver`	Task loss plus terminal-state consistency across different step sizes or numerical solvers.	Tests whether numerical reparameterization stability is beneficial.
`jac`	Task loss plus local Jacobian smoothness or spectral smoothness regularization.	Tests whether smoothing local propagation operators improves internal field quality or generalization.
`full`	Task loss plus reveal consistency, solver consistency, and Jacobian regularization.	Tests whether the complete field-aware objective improves both internal propagation quality and downstream generalization.

A.6.2. Objective Ablations

We compare several training objectives to isolate which component of the field-aware objective contributes to path consistency, numerical stability, or generalization. Table 9 summarizes the ablation design.

A.6.3. Real and Semi-Real Task Transfer

We evaluate reveal-path and field-aware objectives on both semi-real and real tasks:

- **Semi-real task:** reveal-path experiments on `sklearn` Digits.
- **Real vision tasks:** CIFAR-100 with patch reveal, and Tiny-ImageNet with patch or frequency reveal.
- **Real text tasks:** AG News and SST-2 with prefix reveal.
- **Real speech task:** Speech Commands with temporal reveal.

Across all tasks, we record a unified set of metrics: **AccAvg**, **UnseenAcc**, **OODAcc**, **PathSensLogit**, **PathSensHidden**, **cross-path agreement**, and **ECE**. These metrics jointly evaluate endpoint accuracy, unseen-path generalization, robustness to distributional shifts, internal path sensitivity, prediction consistency, and calibration.

A.6.4. Collapse Prevention and Diagnostic Metrics

Path consistency alone does not imply a good model. A degenerate classifier that always predicts a constant class can also achieve low path sensitivity. Therefore, when using field-aware objectives, we jointly monitor accuracy, prediction entropy, class balance, calibration, and cross-path agreement.

This distinction is important because a reduction in path sensitivity can arise from two qualitatively different mechanisms. In the desirable case, the model learns informative path-invariant representations. In the degenerate case, the model collapses to uninformative representations or constant predictions. We therefore interpret path-consistency improvements only together with endpoint performance and anti-collapse diagnostics.

A.7. Training-Algorithm Pareto Analysis and "Anti-Field" Dynamics

Once propagation-field quality is included in the training objective, optimization is no longer governed by a single task loss. Instead, it becomes a multi-objective problem involving task performance, field quality, and computational cost. This group of experiments compares the Pareto behavior of different training algorithms under the same field model class and the same teacher-flow environment.

Table 10. Training algorithms for task–field–compute Pareto analysis.

Algorithm	Training Strategy	Question Tested
FullBPTT	End-to-end backpropagation through the full solver.	Tests whether the standard training method still lies on the Pareto frontier.
SegmentShooting	Segment-wise shooting or collocation with consistency constraints between local segments.	Tests whether local dynamical training improves field recovery.
LocalFieldMatch	Directly matches the local vector field or derivatives.	Tests whether local field supervision is more effective than global endpoint supervision.
PCGrad / MGDA	Applies multi-objective gradient-conflict mitigation.	Tests whether conflicts between task and field gradients require specialized optimization methods.
Curriculum	Trains the field or local dynamics first, then trains the task objective.	Tests whether training order changes the resulting Pareto point.
Alternating	Alternates between task-optimization steps and field-optimization steps.	Tests whether alternating optimization improves the trade-off among objectives.
ProjectedTask	Projects the task gradient onto directions that do not harm the field objective.	Tests whether "anti-field" conflicts can be mitigated through gradient intervention.

A.7.1. Compared Algorithms

Table 10 summarizes the training algorithms compared in this experiment.

A.7.2. Training-Dynamics Logging

For each training epoch, we record task accuracy, trajectory RMSE, derivative RMSE, and reparameterization error. We also compute the cosine similarity between the task gradient and the field gradient:

$$\cos(g_{\text{task}}, g_{\text{field}}) = \frac{\langle g_{\text{task}}, g_{\text{field}} \rangle}{\|g_{\text{task}}\|_2 \|g_{\text{field}}\|_2}. \tag{43}$$

We report the mean cosine similarity, the minimum cosine similarity, and the fraction of epochs or optimization steps with negative cosine values. These statistics are used to determine whether task–field conflicts emerge in later stages of training. Finally, we compare the algorithms along three axes: task performance, field recovery, and computational cost, thereby characterizing their task–field–compute Pareto trade-offs.

A.8. Field Preservation Regularization (FPR) in Continual Learning

The FPR experiments apply the propagation-field perspective to continual learning. The central question is whether catastrophic forgetting can be interpreted not only as degradation of endpoint performance, but also as the rewriting of internal propagation fields associated with previous tasks. We therefore ask whether preserving the hidden trajectories and local Jacobian fields of old tasks can reduce forgetting or provide an additional dimension of retention beyond accuracy.

A.8.1. Dataset and Task Split

We use Split CIFAR-100 as the continual learning benchmark. The dataset is divided into 20 sequential tasks, with each task containing 5 classes. Models are trained task by task. After finishing task t, we evaluate the model on all tasks observed so far.

The baseline diagnostic setting is pure sequential fine-tuning, without replay and without regularization-based protection. The final outputs of the experimental pipeline include the accuracy matrix, drift records, correlation reports, and phase-wise

Table 11. Compared methods in the FPR continual learning experiments.

Method	Core Mechanism	Notes
Finetune	Sequential training without protection.	Forgetting baseline.
EWC	Fisher-weighted parameter-space L_2 regularization.	Protects parameters.
LwF	Distills logits from the old model.	Protects functional outputs, with no or weak sample storage.
ER	Stores a sample buffer and trains with cross-entropy on mixed old and new samples.	Replay baseline.
DER++	Stores samples, old logits, and labels; aligns old logits using MSE together with cross-entropy.	Strong replay/distillation baseline.
FPR-Traj	Preserves hidden trajectories of the old and new models on anchor samples.	Constrains state propagation only.
FPR-Jac	Preserves local Jacobian fields of the old and new models.	Constrains local propagation operators only.
FPR-Full	Preserves both hidden trajectories and local Jacobian fields.	Full field-preservation variant.

comparison tables.

A.8.2. Model: FieldResNet-18

The backbone is a ResNet-18-style architecture, modified to explicitly expose hidden states from multiple intermediate layers. In our experiments, we record 6 intermediate layers covering early, middle, and late depths of the network.

For each anchor sample, we record the hidden trajectory and a local Jacobian proxy under both the old model and the new model. This allows us to compare how the same input is propagated internally before and after subsequent task learning.

A.8.3. Compared Methods

Table 11 summarizes the compared continual learning methods.

A.8.4. Key Evaluation Metrics

Table 12 summarizes the key evaluation metrics.

A.8.5. Phase 0: Pure Forgetting Diagnostics

In Phase 0, we train the model sequentially on all 20 tasks using the Finetune baseline. After each task is completed, we evaluate the model on all previously observed tasks and generate an accuracy matrix.

For each old-task anchor sample, we record parameter drift, trajectory drift, Jacobian drift, and accuracy drop. The phase outputs include `accuracy_matrix.json`, `drift_records.csv`, `correlation_report.json`, `p0_drift_heatmap.png`, and `p0_drift_scatter.png`. The goal of this phase is diagnostic: to determine which types of drift are more closely associated with forgetting, rather than to propose a new method.

A.8.6. Phase 1: Full Comparison of Eight Methods

In Phase 1, we run Finetune, EWC, LwF, ER, DER++, FPR-Traj, FPR-Jac, and FPR-Full under the same task order, model architecture, and evaluation protocol. We record the complete accuracy matrix and report AA, BWT, FWT, FRS, and JRS.

The phase outputs include `results.json` and `p1_comparison.png`. This phase compares traditional continual learning methods and FPR variants in terms of both function retention and field retention.

Table 12. Evaluation metrics for FPR continual learning experiments.

Metric	Definition	Interpretation
AA	Average accuracy of the final model over all tasks.	Overall continual learning performance.
BWT	Change in old-task accuracy from immediately after learning the task to the final model.	Negative values indicate forgetting; values closer to 0 are better.
FWT	Initial performance on a task before learning it, relative to a random baseline.	Measures forward transfer.
FRS	Difference between hidden trajectories of the old and new models on anchor samples.	Lower values indicate better representation-trajectory retention.
JRS	Difference between local Jacobian proxies of the old and new models on anchor samples.	Lower values indicate better retention of local propagation operators.
FRS_final / JRS_final	Field-retention metrics comparing the final model with previous task-specific models.	Used to analyze long-term field retention.

A.8.7. Phase 2: FPR Component and Layer Ablations

Phase 2 performs component and layer-level ablations. The component ablation compares TrajOnly, JacOnly, and Full variants. The layer ablation compares field preservation applied to early layers, middle layers, late layers, and all selected layers.

We compare the effects of different FPR components and protected layer groups on AA, BWT, FRS, and JRS. The phase outputs include `p2_ablation.png` and `results.json`. This phase is used to determine which objects and which layers should be protected for effective propagation-field retention.

A.8.8. Phase 3: Budget-Efficiency Pareto Experiments

Phase 3 studies the budget-efficiency trade-off. We vary the anchor or memory budget, for example 50, 100, 200, and 500 samples per task. Under each budget, we compare Finetune, ER, DER++, and FPR-Full using AA, BWT, FRS, and JRS.

The phase outputs include `budget_results.json` and `p3_budget.png`. This phase examines whether FPR can provide advantages under low-budget settings, especially when label storage or replay memory is limited, and whether it is complementary to replay-based methods.

A.8.9. Phase 4: Hybrid Enhancement Experiments

In Phase 4, FPR is no longer treated only as a standalone continual learning method. Instead, it is used as a field-retention regularizer that can be added to existing continual learning methods.

We compare ER with ER+FPR-Late, and DER++ with DER+++FPR-Full. Finetune and FPR-Full are retained as the forgetting lower bound and the standalone FPR baseline, respectively. All methods are run under the same memory or anchor budget, with particular focus on small to medium budgets such as 50, 100, and 200 samples per task.

We record AA, BWT, FWT, FRS, JRS, FRS_final, and JRS_final. For each hybrid method, we also compute its improvement relative to the corresponding base method:

$$\Delta\mathrm{AA} = \mathrm{AA}_{\mathrm{hybrid}} - \mathrm{AA}_{\mathrm{base}}, \tag{44}$$

$$\Delta\mathrm{BWT} = \mathrm{BWT}_{\mathrm{hybrid}} - \mathrm{BWT}_{\mathrm{base}}, \tag{45}$$

$$\Delta\mathrm{FRS} = \mathrm{FRS}_{\mathrm{hybrid}} - \mathrm{FRS}_{\mathrm{base}}, \tag{46}$$

and

$$\Delta\mathrm{JRS} = \mathrm{JRS}_{\mathrm{hybrid}} - \mathrm{JRS}_{\mathrm{base}}. \tag{47}$$

This phase evaluates whether field preservation can improve existing replay or distillation methods by adding an internal-propagation retention objective.

Table 13. **PDE prototype experiments.** Controlled PDE systems with known propagation laws are used to compare endpoint maps, time-conditioned maps, propagation generators, and structured propagation models. Metrics test horizon extrapolation, step-size reparameterization, energy evolution, and perturbation propagation. Lower MSE/regrid/perturbation error is better; higher energy correlation is better.

System	Model	Endpoint MSE	$T = 2.0$ MSE	Regrid $dt/2$	Energy r	Perturb. MSE
PDE-A: Linear Adv-Diff	M1 Endpoint	2.529e-03	1.217e+00	–	–	–
	M2 TimeCond	2.506e-01	1.394e+00	–	–	–
	M3 Generator	8.220e-01	1.933e+00	**1.772e-03**	0.986	**7.287e-03**
	M4 Structured	1.748e-01	**1.948e-01**	2.170e-02	**0.995**	7.628e-03
PDE-B: Nonlinear ADR	M1 Endpoint	4.638e-03	4.056e-01	–	–	–
	M2 TimeCond	8.382e-02	7.337e-01	–	–	–
	M3 Generator	3.214e-01	7.846e-01	**6.558e-03**	**0.998**	**5.547e-03**
	M4 Structured	1.174e-01	**9.944e-02**	2.980e-02	0.976	7.500e-03
PDE-C: Non-autonomous	M1 Endpoint	3.832e-03	5.024e-01	–	–	–
	M2 TimeCond	1.113e-01	7.990e-01	–	–	–
	M3 Generator	5.598e-01	9.267e-01	**8.443e-03**	0.977	**8.817e-03**
	M4 Structured	1.260e-01	**9.195e-02**	2.494e-02	**0.987**	9.372e-03
PDE-B controls	M3 NC1 rand-pair	1.712e-01	–	1.703e-01	–	–
	M3 NC2 shuffled	1.376e+00	–	1.391e+00	–	–
	M3 normal	3.214e-01	–	3.207e-01	–	–

Message: endpoint fitting can match a training horizon, but stable propagation recovery requires generator-style or structured propagation models.

A.9. Additional Experiment Result

Table 14. **Propagation-field extraction and controlled teacher-flow experiments.** Ordinary networks are analyzed by treating depth as pseudo-time. Teacher-flow experiments use a latent ODE with accessible ground-truth trajectories and derivatives, allowing direct comparison between endpoint accuracy and field recovery.

Block	Setting / Model	Acc.	NormPath	Curv.	VelAlign	Traj./Spectral	Deriv./Jac.
ResNet depth	Depth 4	1.0000	1.1936	0.0480	0.9873	–	–
	Depth 6	1.0000	1.2982	0.0822	0.9705	–	–
	Depth 12	1.0000	1.4656	0.2177	0.9173	–	–
Architecture	ResNet	1.0000	1.2803	0.0885	0.9702	–	–
	Transformer	1.0000	1.3418	0.0816	0.9709	Procrustes 1.3968	–
Jacobian spectra	Adjacent-layer WDist	–	–	–	–	mean 0.00833	min 0.0029 / max 0.0168
	Depth refinement	1.0000	–	–	–	hidden/logit err. 10^{-6}–10^{-5}	–
Teacher A	M1 EndpointMLP	0.9975	–	–	–	–	–
	M2 FlowEndOnly	0.9975	6.5657	–	–	Traj. 3.5229	Deriv. 2.7627 / JacW 0.2505
	M3 FlowFieldLoss	0.9975	**1.6595**	–	–	**Traj. 0.0590**	**Deriv. 0.1495 / JacW 0.0409**
Teacher B	M1 EndpointMLP	0.9937	–	–	–	–	–
	M2 FlowEndOnly	0.9900	4.0904	–	–	Traj. 3.3263	Deriv. 1.8882 / JacW 0.2170
	M3 FlowFieldLoss	0.9900	**1.6551**	–	–	**Traj. 0.0694**	**Deriv. 0.1632 / JacW 0.0333**

Message: ordinary deep networks expose measurable depth-wise geometry; in teacher-flow systems, identical endpoint accuracy can hide large differences in true field recovery.

Table 15. **Real and semi-real reveal-path transfer.** The same object is revealed through multiple paths: image patch/frequency reveal, text prefix/evidence reveal, and speech temporal-window reveal. Metrics jointly test endpoint accuracy, unseen-path generalization, OOD robustness, path sensitivity, and calibration.

Dataset	Obj.	AccAvg	Unseen	OOD	PathSensLogit	PathSensHidden	ECE
digits	task	**0.7528**	**0.6633**	**0.8089**	8.4402	6.2641	**0.0325**
	reveal	0.5261	0.4222	0.5978	**3.5193**	2.1058	0.1746
	full	0.1894	0.1833	0.1844	10.4550	**2.0529**	0.5876
CIFAR-100	task	0.0405	0.0295	0.0570	15.1867	7.2761	0.0473
	reveal	**0.0558**	**0.0490**	**0.0600**	7.2521	3.3842	0.0239
	full	0.0093	0.0090	0.0100	**3.0355**	**2.1391**	**0.0229**
Tiny-ImageNet	task	0.0970	0.0853	0.1000	4.4924	4.4374	0.1018
	reveal	0.0943	0.0853	0.1053	1.7353	2.3665	0.0539
	full	**0.1180**	**0.1087**	**0.1280**	**1.2742**	**2.0651**	**0.0182**
AG News	task	**0.3642**	**0.3639**	**0.2753**	**0.0470**	**0.1387**	**0.0073**
	reveal	0.2804	0.2802	0.2603	12.0342	4.0788	0.0337
	full	0.2500	0.2500	0.2505	0.1302	0.1644	0.2182
SST-2	task	**0.5789**	**0.5792**	0.5261	**0.0078**	0.1158	**0.0045**
	reveal	0.5379	0.5380	0.5409	4.3906	0.4716	0.0250
	full	0.5572	0.5572	**0.5572**	0.0592	**0.1010**	0.0992
Speech	task	**0.1925**	**0.1917**	0.1298	0.2381	0.3804	0.0465
	reveal	0.1866	**0.1917**	**0.1386**	0.1071	0.1591	0.0547
	full	0.1121	0.1121	0.1091	**0.0682**	**0.1331**	**0.0356**

Message: field-aware objectives help when aligned with task structure, but over-regularization can reduce PathSens while collapsing accuracy.

Table 16. **Current strict-suite summary: task–field–compute tradeoffs and regime dependence.** For task-structure and phase-diagram rows, Δ denotes the field-aware objective minus the task-only baseline; negative ΔField means lower field error. For Pareto rows, Task is TestAcc and Field reports Traj./Deriv. RMSE. The results show that field-aware learning is useful only in specific path-structured regimes and that optimization methods occupy different task–field–compute tradeoff points.

Block	Setting / Method	Task Axis	OOD Axis	Field Axis	Conflict / Selector	Takeaway
Task structure	$\lambda = 0.0$ endpoint	ΔID -0.530	ΔOOD -0.115	ΔTraj -1.925 / ΔDeriv -1.830	–	field improves, task collapses
	$\lambda = 0.5$ medium path	ΔID -0.070	ΔOOD $+0.210$	ΔTraj -0.811 / ΔDeriv -0.031	–	OOD gain despite ID drop
	$\lambda = 0.75$ strong path	ΔID $+0.015$	ΔOOD $+0.075$	ΔTraj -0.260 / ΔDeriv -0.722	–	best aligned regime
	$\lambda = 1.0$ long path	ΔID -0.185	ΔOOD -0.030	ΔTraj -0.840 / ΔDeriv -0.643	–	over-constrained
Pareto training	FullBPTT	Acc. 0.590	–	Traj 1.053 / Deriv 2.674	neg-grad 0.18	baseline point
	LocalFieldMatch	Acc. 0.615	–	Traj 1.034 / Deriv 2.655	neg-grad 0.36	better field and accuracy
	MGDA	Acc. 0.665	–	Traj 1.297 / Deriv 2.830	neg-grad 0.55	best task accuracy
	Curriculum	Acc. 0.660	–	Traj 1.115 / Deriv 2.735	neg-grad 0.00	stable compromise
Anti-field dynamics	FullBPTT	Acc. 0.590	–	Traj 1.053 / Deriv 2.674	cos 0.405, neg 0.18	task–field conflict emerges
	ProjectedTask	Acc. 0.670	–	Traj 1.305 / Deriv 2.831	cos -0.025, neg 0.55	task improves, field worsens
Phase diagram	$\lambda = 0.0$, avg.	ΔID -0.329	ΔOOD -0.323	ΔTraj -0.660 / ΔDeriv -0.219	joint $0/16$	endpoint-like over-regularization
	$\lambda = 0.5$, avg.	ΔID $+0.137$	ΔOOD $+0.064$	ΔTraj -0.567 / ΔDeriv -0.088	joint $8/16$	main positive window
	$\lambda = 1.0$, avg.	ΔID -0.031	ΔOOD -0.087	ΔTraj -0.190 / ΔDeriv -0.034	joint $0/16$	strong path alone insufficient
Multivariate	Global predictors of ΔOOD	$\mathrm{corr}(\lambda, \Delta\mathrm{OOD}) = 0.461$	–	$\mathrm{corr}(\Delta\mathrm{Deriv}, \Delta\mathrm{OOD}) = 0.556$	RF: Deriv 0.363, λ 0.347, Jac 0.210	path structure and local field matter
	Stratified field proxy	–	low-λ: $-0.33/-0.34$	high-λ: $+0.32/+0.29$	–	field quality is conditionally useful

Message: field-aware learning is not a universal accuracy booster. It reduces field errors broadly, but improves ID/OOD only when aligned with path structure and appropriate optimization; otherwise it can over-regularize or trade task performance for field quality.

Table 17. **Continual learning with field preservation.** Split CIFAR-100 is divided into 20 sequential tasks. Phase 0 diagnoses whether field drift correlates with forgetting; Phases 1–3 evaluate standalone FPR and its budget behavior; Phase 4 adds FPR as a field-retention regularizer to replay/distillation methods. Higher AA and BWT are better; lower FRS/JRS are better. In Phase 4, positive ΔAA/ΔBWT and negative ΔFRS/ΔJRS indicate improvement.

Phase	Setting	Method / Test	Function metric	Field metric
0	Drift diagnosis	Trajectory drift vs. drop	$r = 0.1748,\ p = 0.0158$	$\rho = 0.1498,\ p = 0.0391$
		Jacobian drift vs. drop	$r = 0.0266,\ p = 0.7159$	$\rho = 0.0070,\ p = 0.9235$
		Parameter drift vs. drop	$r = -0.0853,\ p = 0.2418$	$\rho = -0.0924,\ p = 0.2046$
1	Method comparison	Finetune	AA 0.0384, BWT -0.6437	FRS 5.7340, JRS 0.1193
		EWC	AA 0.0140, BWT -0.0921	FRS 7.2487, JRS 0.1195
		LwF	AA 0.0467, BWT -0.6925	FRS 1.5442, JRS 0.1183
		ER	AA 0.1160, BWT -0.4955	FRS 0.7644, JRS 0.1610
		DER++	AA $\mathbf{0.1346}$, BWT -0.5248	FRS 0.9209, JRS 0.1768
		FPR-Traj	AA 0.0374, BWT -0.6546	FRS 0.5568, JRS $\mathbf{0.0332}$
		FPR-Jac	AA 0.0345, BWT -0.5707	FRS 1.2716, JRS 0.0536
		FPR-Full	AA 0.0360, BWT -0.6263	FRS 1.5758, JRS 0.0404
2	FPR ablation	FPR-Full	AA 0.0353, BWT -0.6360	FRS $\mathbf{0.4083}$, JRS $\mathbf{0.0347}$
		FPR-TrajOnly	AA 0.0368, BWT -0.6534	FRS 13.5703, JRS 0.1777
		FPR-JacOnly	AA 0.0334, BWT -0.5735	FRS 1.9565, JRS 0.0539
		FPR-Early	AA 0.0289, BWT -0.6182	FRS 2.0783, JRS 0.0743
		FPR-Mid	AA 0.0382, BWT -0.6582	FRS 3.4924, JRS 0.0780
		FPR-Late	AA 0.0339, BWT -0.5223	FRS 4.3175, JRS 0.1226
3	Budget	50 samples/task	AA: ER 0.0584, DER++ 0.0712, FPR 0.0367	JRS: ER 0.1715, DER++ 0.1675, FPR $\mathbf{0.0447}$
		100 samples/task	AA: ER 0.0897, DER++ 0.1080, FPR 0.0375	JRS: ER 0.1552, DER++ 0.2443, FPR $\mathbf{0.0500}$
		200 samples/task	AA: ER 0.1156, DER++ 0.1308, FPR 0.0321	JRS: ER 0.2074, DER++ 0.2062, FPR $\mathbf{0.0448}$
		500 samples/task	AA: ER 0.1596, DER++ 0.1903, FPR 0.0361	JRS: ER 0.2391, DER++ 0.2132, FPR $\mathbf{0.0600}$
4	Hybrid FPR	DER++ $\rightarrow$ DER++ + FPR-Full, 50	ΔAA -0.0021, ΔBWT -0.0218	ΔFRS -0.2579, ΔJRS -0.1279
		DER++ $\rightarrow$ DER++ + FPR-Full, 100	ΔAA $+0.0008$, ΔBWT -0.0429	ΔFRS -0.1135, ΔJRS -0.1176
		DER++ $\rightarrow$ DER++ + FPR-Full, 200	ΔAA $+\mathbf{0.0111}$, ΔBWT $+\mathbf{0.0571}$	ΔFRS $-\mathbf{0.2661}$, ΔJRS $-\mathbf{0.0748}$
		ER $\rightarrow$ ER + FPR-Late, 50	ΔAA -0.0267, ΔBWT -0.0581	ΔFRS $+0.5128$, ΔJRS $+0.0012$
		ER $\rightarrow$ ER + FPR-Late, 100	ΔAA -0.0470, ΔBWT -0.1027	ΔFRS $+0.4941$, ΔJRS $+0.0028$
		ER $\rightarrow$ ER + FPR-Late, 200	ΔAA -0.0461, ΔBWT -0.1415	ΔFRS $+0.3111$, ΔJRS -0.0207

Message: standalone FPR strongly preserves propagation-field metrics, especially JRS, but does not replace replay or distillation for function retention. As a regularizer, FPR is most effective when combined with a strong base method: DER++ + FPR-Full at 200 samples/task improves AA, BWT, FRS, and JRS simultaneously.