Second-Order Structure in Learned Optimizers: Evidence for Low-Dimensional Manifolds of Learning Rules

Het Patel
hpatel54@binghamton.com

Abstract

Meta-learning systems that learn to optimize other models—"learned optimizers"—offer a concrete setting in which to study higher-order structure in learning dynamics. This work asks a focused question: *when we meta-train learned optimizers across families of tasks and random seeds, do their parameters converge to a single attractor, or to a lower-dimensional manifold of quasi-equivalent learning rules?* We construct an experimental framework in which a small neural optimizer is meta-trained to solve 1D regression tasks (random sinusoids, polynomials, exponentials) for a fixed base model, and analyze the resulting learned optimizer parameters across seeds using distance statistics, principal component analysis (PCA), and perturbation experiments. In our setting, final learned optimizers from different seeds do *not* collapse to a single point: pairwise distances remain large, with a mean of approximately 1.71 in an 81-dimensional parameter space. Nevertheless, they exhibit strong *dimension collapse*: roughly six principal components explain over 90% of the variance in the learned optimizer parameters. Perturbation experiments around a reference optimizer show weakly repulsive or neutral behavior rather than strong local contraction, suggesting the absence of a robust point attractor at that location. Taken together, these results support a "manifold" or "valley" picture in which meta-learning carves out a low-dimensional region of viable learning rules in optimizer parameter space, onto which different runs collapse but within which they can drift. We also view this project as a case study in *AI-accelerated science*: the hypothesis, experimental design, and analysis plan were defined by the author, while implementation was accelerated using large language model (LLM)-based code generation with manual verification of core statistics. Finally, we outline follow-up experiments that track dimension collapse over training time and couple meta-learning with dynamic capacity mechanisms, with an eye toward future self-improving systems that navigate manifolds of learning rules rather than searching arbitrary algorithm space.

Keywords

meta-learning, learned optimizers, dynamical systems, low-dimensional manifolds, attractors, representation collapse, AI-accelerated research

Preprint,

ACM Reference Format:
Het Patel. 2025. Second-Order Structure in Learned Optimizers: Evidence for Low-Dimensional Manifolds of Learning Rules. In *Proceedings of Preprint*. ACM, New York, NY, USA, 6 pages.

1 Introduction

Gradient-based learning is usually analyzed at the level of a model's parameters: given an objective $L(\theta)$, one specifies an update rule such as stochastic gradient descent (SGD) and studies the resulting dynamics in parameter space. Meta-learning, and learned optimizers in particular, push this one level higher: the object of interest is the *learning rule* itself, parameterized by α and trained to minimize a meta-objective over tasks [1, 2].

This shift naturally raises a dynamical-systems question:

> As we meta-train a learned optimizer across many tasks and random seeds, do its parameters α converge to a *single* attractor, or to a *lower-dimensional manifold* of quasi-equivalent learning rules?

A "single attractor" picture would suggest that, for a given task family and base model, there is essentially one optimal second-order rule, and that perturbations around this rule are pulled back by the meta-training dynamics. A "manifold" picture is weaker but still structured: different runs end at different points, but those points lie on a low-dimensional surface in the high-dimensional optimizer parameter space, and effective dynamics are constrained to that surface.

Beyond its intrinsic scientific interest, this question touches on broader issues in *automated machine learning* and *self-improving AI*. If useful learning rules occupy a low-dimensional manifold inside a much larger parameter space, then an agent that can navigate this manifold may be able to adapt its own learning behavior efficiently, rather than searching over arbitrary algorithm designs. Understanding the geometry of the space of optimizers is therefore relevant not only for meta-learning, but also for future systems that explicitly optimize over their own learning rules.

In this paper we take an empirical first step toward answering this question in a deliberately simple setting. We build a small experimental framework in which:

- The *inner loop* learns to solve 1D regression tasks (random sinusoids, low-degree polynomials, exponentials) using a small multilayer perceptron (MLP) base model.
- The *outer loop* meta-learns a parametric optimizer: a small neural network that maps per-parameter gradients, momenta, and a time feature to parameter updates.
- We train this learned optimizer from scratch for multiple random seeds under identical settings and analyze the resulting parameter vectors α.

We conduct three primary experiments: **(A)** a convergence experiment across seeds; **(B)** a manifold analysis via PCA; and **(C)** a

perturbation experiment that probes local attractor behavior around a reference optimizer.

Contributions. Our contributions are:

- We propose a simple experimental setup for probing attractor structure in the space of learned optimizer parameters.
- In this setup, we provide empirical evidence that learned optimizers trained from different seeds lie on a low-dimensional manifold (six principal directions explaining > 90% of variance) rather than collapsing to a single point.
- We show that small perturbations around a reference optimizer do not contract back, arguing against strong local point-attractor behavior at that location.
- We discuss broader implications for AutoML and self-improving systems: if learning rules live on a low-dimensional manifold, future agents may be able to optimize over *optimizer manifolds* instead of searching the full algorithm space.
- We explicitly document an *AI-accelerated* research workflow in which the author provides the hypothesis, experimental design, and verification strategy, while LLM-based code generation accelerates implementation, and we argue that this "director" role is itself an important research skill.

2 Background and Related Work

2.1 Meta-learning and learned optimizers

Meta-learning aims to train models that quickly adapt to new tasks by exploiting experience over a distribution of tasks [7, 8]. One influential line of work parameterizes the optimizer itself as a neural network [1, 2, 4], often applied coordinate-wise to model parameters.

Given a task family $\{\tau\}$ and an inner model f_θ, such systems meta-learn an update rule

$$\theta_{t+1} = \theta_t + g_\alpha\big(\nabla_{\theta_t}\ell_\tau(\theta_t), s_t\big), \tag{1}$$

where g_α is the learned optimizer and s_t denotes optimizer state (e.g., momentum). The meta-objective typically minimizes validation loss after a fixed inner training budget:

$$J(\alpha) = \mathbb{E}_{\tau\sim p(\tau)}\left[L_\tau^{\text{val}}(\theta_T(\alpha;\tau))\right], \tag{2}$$

and is optimized by backpropagating through the unrolled inner optimization.

Recent work has demonstrated that learned optimizers can scale and generalize to a range of tasks and architectures [2, 3], and there is growing interest in reverse-engineering their behavior to understand what learning rules they implement in practice.

2.2 Loss landscapes and manifolds of solutions

The geometry of loss landscapes in deep networks is highly structured: seemingly different minima are often connected by low-loss paths [5, 6], and solution sets can form extended flat regions or valleys rather than isolated points. Dimension-reduction techniques such as PCA and low-rank subspace methods have been used to show that ensembles of independently trained models often lie near low-dimensional subspaces in parameter space.

This perspective has motivated methods that explicitly optimize in learned subspaces or parameterizations. However, relatively little work has directly examined *dimension collapse in the space of optimizer parameters themselves*, treating the meta-parameters α as the state of a dynamical system.

2.3 Attractors and dynamical systems

In a discrete-time dynamical system $x_{t+1} = F(x_t)$, an *attractor* is an invariant set A whose basin of attraction captures the long-term behavior of trajectories. Attractors may be fixed points, limit cycles, or more complex sets such as strange attractors.

In the context of meta-learning, the learned optimizer parameters α_t evolve under an effective update rule

$$\alpha_{t+1} = \Phi(\alpha_t) = \alpha_t - \eta\nabla_\alpha J_t(\alpha_t), \tag{3}$$

where J_t encodes the sampled meta-loss at step t and η is a meta-learning rate. It is therefore natural to ask whether there are attractors in α-space, what their geometry is, and whether they are point-like or low-dimensional manifolds.

3 Experimental Setup

We now describe our controlled setting for probing attractor structure in learned optimizer parameter space.

3.1 Task distribution: 1D regression

We define a distribution over 1D regression tasks. Each task τ samples a function $f_\tau : \mathbb{R} \to \mathbb{R}$ from one of three families:

Sinusoids.

$$f(x) = A\sin(\omega x + \phi) + b, \tag{4}$$

with amplitudes $A \sim \mathcal{U}(0.5, 2.0)$, frequencies $\omega \sim \mathcal{U}(0.5, 3.0)$, phases $\phi \sim \mathcal{U}(0, 2\pi)$, and offsets $b \sim \mathcal{U}(-0.5, 0.5)$.

Polynomials.

$$f(x) = \sum_{i=0}^{d} a_i x^i, \tag{5}$$

where $d \in \{2, 3\}$ and coefficients $a_i \sim \mathcal{U}(-1.0, 1.0)$.

Exponentials.

$$f(x) = Ae^{\beta x} + c, \tag{6}$$

with $A \sim \mathcal{U}(0.3, 1.0)$, $\beta \sim \mathcal{U}(-0.5, 0.5)$, and $c \sim \mathcal{U}(-0.5, 0.5)$.

For each task we sample:

- Training inputs x_{train} uniformly from $[-2, 2]$, and add Gaussian noise to outputs.
- Validation inputs x_{val} as a fixed grid on $[-2, 2]$, with noiseless outputs.

We normalize inputs by $x \mapsto x/2$ so that $x \in [-1, 1]$, and standardize outputs using the mean and standard deviation of y_{train}. The inner objective is mean squared error on standardized outputs.

3.2 Base model (inner learner)

The base model is a small MLP acting on scalar inputs. In our main experiments we use a 1–32–1 architecture with tanh activations:

$$h = \tanh(W_1 x + b_1), \tag{7}$$

$$\hat{y} = W_2 h + b_2, \tag{8}$$

with $W_1 \in \mathbb{R}^{32\times 1}$, $b_1 \in \mathbb{R}^{32}$, $W_2 \in \mathbb{R}^{1\times 32}$, and $b_2 \in \mathbb{R}$. Weights are initialized with scaled Gaussian noise and biases are initialized to zero.

We represent parameters θ as a list of tensors and implement the forward pass using explicit linear operations so that we can easily flatten and unflatten θ when applying the learned optimizer.

3.3 Learned optimizer architecture

The learned optimizer g_α is applied element-wise to base model parameters. For each scalar parameter θ_i we maintain a momentum-like state m_i and compute an update $\Delta\theta_i$ from:

- the current gradient $g_i = \partial\ell/\partial\theta_i$,
- the current momentum m_i,
- a scalar time feature $\log(t+1)$, where t is the inner step index.

We form a 3D input vector $[g_i, m_i, \log(t+1)]$ and feed it through a small MLP

$$\Delta\theta_i = \mathrm{MLP}_\alpha\big([g_i, m_i, \log(t+1)]\big), \tag{9}$$

where $\mathrm{MLP}_\alpha : \mathbb{R}^3 \to \mathbb{R}$ has one hidden layer of width 16 with tanh activation. Weights are initialized with small Gaussian noise and biases to zero to avoid large early updates.

For a base model with P parameters, we flatten gradients and momenta into vectors of length P, apply the MLP in batch, and reshape the resulting update vector back to the original parameter shapes. Momentum is updated as

$$m_i^{(t+1)} = \beta m_i^{(t)} + (1-\beta) g_i^{(t)}, \quad \beta = 0.9. \tag{10}$$

The full learned-optimizer parameter vector α consists of all weights and biases of MLP_α. In our configuration this yields

$$\alpha \in \mathbb{R}^{81}. \tag{11}$$

3.4 Inner loop and meta-objective

Given a task τ and a learned optimizer g_α, the inner loop starts from a random initialization θ_0 and performs T update steps:

$$\theta_{t+1} = \theta_t + \Delta\theta_t, \tag{12}$$

$$\Delta\theta_t = g_\alpha\big(\nabla_{\theta_t} \ell_\tau^{\mathrm{train}}(\theta_t), m_t, t\big), \tag{13}$$

with momentum updated as above. The inner loss is mean squared error on y_{train}.

After T steps we evaluate the validation loss

$$L_\tau^{\mathrm{val}}(\alpha) = \ell_\tau^{\mathrm{val}}\big(\theta_T(\alpha;\tau)\big). \tag{14}$$

The meta-objective over a batch of tasks is

$$J(\alpha) = \mathbb{E}_{\tau\sim p(\tau)}\left[L_\tau^{\mathrm{val}}(\alpha)\right], \tag{15}$$

approximated via Monte Carlo sampling over tasks. We differentiate $J(\alpha)$ with respect to α by backpropagating through the unrolled inner optimization. The outer optimizer is Adam with learning rate 10^{-3} and gradient clipping.

3.5 Hyperparameters and training regimes

We consider two regimes:

DEBUG regime. Intended for quick iteration:

- meta-steps: 100,
- inner steps: 10,
- meta-batch size: 4 tasks,
- seeds: 3.

FULL regime. Used for the analyses reported in this paper:

- meta-steps: 1000,
- inner steps: 30,
- meta-batch size: 16 tasks,
- seeds: 10.

Unless otherwise stated, results refer to the FULL regime.

4 Experiments and Results

4.1 Experiment A: Convergence across seeds

4.1.1 Objective. Assess whether meta-training from different random initializations of α converges to a single point in optimizer parameter space, or to a more extended region.

4.1.2 Procedure. For each of $S = 10$ seeds, we initialize $\alpha_0^{(s)}$ randomly and run 1000 meta-steps in the FULL regime. At the end of training we extract the flattened parameters

$$\alpha^{(s)} \in \mathbb{R}^{81}, \quad s = 1, \ldots, S. \tag{16}$$

We compute:

- the norm $\|\alpha^{(s)}\|_2$ for each seed;
- pairwise distances

$$d_{s,s'} = \|\alpha^{(s)} - \alpha^{(s')}\|_2 \quad (s \neq s');$$

- descriptive statistics (mean, standard deviation) over $\{d_{s,s'}\}$.

We also record meta-training loss $J_t^{(s)}$ over time.

4.1.3 Results. In the FULL regime we observe:

- The mean pairwise distance between final learned optimizers is approximately

$$\mathbb{E}_{s\neq s'}[d_{s,s'}] \approx 1.71,$$

  with a spread on the order of a few tenths.
- The norms of the final optimizers are all of comparable magnitude (on the order of 1), indicating that the parameters lie on a relatively thin shell in $\mathbb{R}^{81}$.
- Meta-loss trajectories generally decrease over time for all seeds, but exhibit noisy fluctuations and some seed-to-seed variability.

For points uniformly distributed on a sphere of radius r in high dimension, typical pairwise distances concentrate near $\sqrt{2}r$. With r in the range observed for $\|\alpha^{(s)}\|_2$, $\sqrt{2}r$ is close to 1.7, which matches the empirically observed mean pairwise distance. This suggests that, rather than collapsing to a tight ball around a single optimum, the final optimizers occupy a roughly shell-like region in parameter space.

We therefore conclude that, in this setting, meta-training from different seeds does *not* converge to a single point or very small neighborhood in α-space.

4.2 Experiment B: Manifold and PCA analysis

4.2.1 Objective. Determine whether the ensemble of final learned optimizers lies close to a lower-dimensional subspace, indicative of dimension collapse in optimizer space.

4.2.2 Procedure. We construct a data matrix $A \in \mathbb{R}^{S\times P}$ whose rows are the flattened learned optimizers:

$$A_{s,:} = (\alpha^{(s)})^\top, \quad S = 10,\ P = 81. \tag{17}$$

We center A by subtracting the column-wise mean and perform PCA on the centered matrix. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_P$ denote the eigenvalues of the empirical covariance matrix. We compute the explained variance ratios

$$\rho_k = \frac{\lambda_k}{\sum_{j=1}^{P} \lambda_j}, \tag{18}$$

and the cumulative variance

$$R_K = \sum_{k=1}^{K} \rho_k. \tag{19}$$

We define the intrinsic dimension at 90% variance as

$$D_{90} = \min\{K : R_K \geq 0.9\}. \tag{20}$$

4.2.3 Results. In the FULL regime, PCA yields:

- The first principal component explains roughly 27.5% of the variance.
- The second principal component explains about 21.9%; together, PC1 and PC2 account for approximately 49.4% of the variance.
- Principal components 3–6 cumulatively raise the explained variance above 90%.

Thus,

$$D_{90} \approx 6,$$

meaning that six principal components suffice to explain at least 90% of the variance among the 81-dimensional learned optimizer parameters.

A scree-style view of cumulative variance (not shown here) displays a sharp rise over the first few components followed by a plateau, consistent with a highly anisotropic distribution whose support is concentrated near a low-dimensional subspace.

These results provide direct evidence for dimension collapse in the space of learned optimizer parameters: although α has 81 degrees of freedom, most of the variability across seeds lies in a subspace of dimension on the order of six.

4.3 Experiment C: Perturbation and local attractor behavior

4.3.1 Objective. Test whether a particular learned optimizer acts as a local attractor: if we perturb it and resume meta-training, do nearby parameter vectors converge back toward it?

4.3.2 Procedure. We select a reference optimizer α (the final optimizer from one chosen seed) and consider perturbation scales $\sigma \in \{0.1, 0.5, 1.0, 2.0\}$. For each σ we perform multiple trials:

(1) Sample a random direction u with $\|u\|_2 = 1$.
(2) Initialize
$$\alpha_0 = \alpha^{+\sigma u}.$$
(3) Resume meta-training for a recovery phase of T_{rec} steps (e.g., 300) with the same outer optimizer and hyperparameters.
(4) At each recovery step t, compute the distance
$$d_t(\sigma) = \|\alpha_t - \alpha^{\|_2}.$$

We average $d_t(\sigma)$ across perturbation trials for each scale.

4.3.3 Results. A representative summary is:

- For $\sigma = 0.1$: $d_0 = 0.10$, final distance $d_T \approx 0.19$ (increase).
- For $\sigma = 0.5$: $d_0 = 0.50$, final distance $d_T \approx 0.54$ (slight increase).
- For $\sigma = 1.0$: $d_0 = 1.00$, final distance $d_T \approx 1.02$ (approximately unchanged).
- For $\sigma = 2.0$: $d_0 = 2.00$, final distance $d_T \approx 1.96$ (slight decrease).

Distance trajectories are generally flat with small drifts, and do not display robust, monotonic contraction for small perturbations. In particular, for $\sigma \in \{0.1, 0.5, 1.0\}$ the distances tend to increase slightly rather than shrink. The mild contraction observed for $\sigma = 2.0$ is small in magnitude and does not constitute strong evidence for a sharply defined basin around α.

We therefore find no sign, in this experiment, that the chosen α is a strong local point attractor: nearby trajectories do not reliably converge back to it under continued meta-training.

5 Discussion

5.1 Manifold of learning rules vs. point attractor

The experiments paint a coherent picture:

- Different random seeds produce learned optimizers that occupy an extended region in parameter space, with pairwise distances comparable to those between random points on a shell (Experiment A).
- Yet this region is strongly low-dimensional: six principal components explain over 90% of the variance among the final optimizers (Experiment B).
- A representative optimizer does not exhibit strong local point-attractor behavior under small perturbations (Experiment C).

These observations are difficult to reconcile with a model in which the meta-objective has a single, sharply defined global optimum in α-space that attracts all trajectories. Instead, they are naturally explained by a *manifold* or *valley* picture: meta-training dynamics collapse onto a low-dimensional region of viable learning rules, but do not select a unique point within that region.

In this view, the low-dimensional manifold encodes a family of quasi-equivalent learning rules—all of which perform well on the task distribution—and stochasticity plus initialization determine which specific rule is reached in a given run. The absence of strong local contraction around a reference point suggests that, once on the manifold, dynamics may be neutral or mildly repulsive along certain directions.

5.2 Relation to representation collapse

The dimension collapse observed here is reminiscent of phenomena in representation learning where high-dimensional activations admit low-dimensional structure after training. In that case, the collapsed object is a *representation of data*; in our case, it is a representation of a *learning rule*.

This suggests an appealing second-order analogy: just as networks learn low-dimensional features of data, meta-learning may discover low-dimensional "features" of useful update rules. Future

work could investigate whether coordinates on the learned optimizer manifold correspond to interpretable properties of learning behavior (e.g., aggressiveness, smoothing, task bias).

5.3 Broader implications: optimizer manifolds and self-improving systems

If learning rules live on a low-dimensional manifold, this has implications for AutoML and self-improving AI systems. A naive view of "learning-to-learn" imagines searching an enormous design space of possible optimization algorithms. Our results suggest a different picture: effective learned optimizers in a given regime may occupy a relatively low-dimensional manifold inside this larger space.

In principle, a sufficiently advanced agent could:

- maintain an internal representation of its location on such an optimizer manifold;
- learn directions on the manifold that improve adaptation to new tasks or shift inductive biases;
- and treat movement along the manifold as a controllable, differentiable way of self-modifying its own learning behavior.

This is analogous to how foundation models navigate latent spaces of natural language or images: they do not search over all possible strings or pixel arrays, but operate in a structured latent geometry. Here, the latent space would be one of *learning rules*, with coordinates capturing families of optimization behaviors. Our small-scale experiments do not realize such a system, but they provide concrete evidence that optimizer manifolds are empirically detectable objects and therefore plausible targets for future self-improving architectures.

6 Limitations and Future Work

6.1 Limitations

This study has several limitations that should temper the interpretation of our results:

- **Toy setting.** Our tasks are 1D regression problems and the base model is a very small MLP. It is unknown whether similar manifold structure appears for higher-dimensional, structured data (e.g., images, language) and larger architectures.
- **Limited sampling.** We use 10 seeds and 1000 meta-steps. More seeds, longer runs, and different outer optimization hyperparameters could reveal additional structure.
- **Single reference point.** The perturbation experiment focuses on one learned optimizer. Other regions of the manifold might exhibit different local behavior.
- **Linear analysis.** PCA captures only linear subspace structure. Nonlinear manifolds would require more sophisticated tools (e.g., manifold learning, nonlinear embeddings) to characterize.

Consequently, our results should be viewed as preliminary evidence for low-dimensional structure in learned optimizer space, not as a complete characterization.

6.2 Temporal dynamics of dimension collapse

The PCA analysis in Experiment B considers only final optimizer states. A natural next step is to study dimension collapse *over the course of meta-training*. Concretely, one could:

- Log $\alpha_t^{(s)}$ at regular intervals and perform PCA at each time t across seeds, tracking an intrinsic dimension estimate $D_{90}(t)$.
- Examine whether $D_{90}(t)$ decreases over training, indicating that dynamics pull parameters into a low-dimensional subspace before primarily moving within it.
- Decompose updates into components parallel and orthogonal to the subspace spanned by late-time principal components, and test whether motion orthogonal to this subspace decays over time.

These experiments would directly test whether meta-learning first contracts trajectories onto a manifold and then explores along it.

6.3 Beyond PCA: basin structure and capacity mechanisms

Several additional directions follow naturally from this work:

Basin structure via clustering. One can cluster optimizer states across time and seeds (e.g., via k-means on $\alpha_t^{(s)}$) to probe coarse basin structure and transitions. We have not performed such clustering in the present study; we propose it as future work to investigate whether trajectories pass through distinct regions (e.g., plateaus, funnels, terminal basins) in α-space.

Dynamic capacity and expand–compress cycles. A central motivating idea is that higher-order "rules of rules" might emerge when models can expand and compress their capacity over a curriculum of tasks. Future experiments could:

- Equip the base model with dynamic capacity (e.g., maskable units or low-rank factors) so that effective dimensionality can grow and shrink.
- Train on a task ladder of increasing complexity (e.g., moving from simple regression to arithmetic operations such as addition, multiplication, and exponentiation).
- Allow expansion when performance saturates and compression when tasks are mastered, regularizing toward minimal sufficient architectures.
- Track how the dimensionality of both θ and α evolve through expand–compress cycles.

Such a setup would let us study how low-dimensional manifolds of learning rules interact with changing model capacity and how catastrophic forgetting manifests when compressing.

7 Conclusion

We have presented an empirical study of second-order structure in the parameter space of a simple learned optimizer. In a controlled 1D regression setting, we find that:

- learned optimizers trained from different seeds do not converge to a single point, but instead occupy an extended region of parameter space;
- this region is strongly low-dimensional, with six principal components explaining over 90% of the variance in an 81-dimensional optimizer parameter vector;

- a representative optimizer does not exhibit strong local point-attractor behavior under small perturbations.

These findings support a manifold-or-valley picture of meta-trained learning rules: meta-learning appears to carve out a low-dimensional region of viable optimizers, rather than a unique optimum. Although our experiments are small in scale, they provide a concrete, falsifiable starting point for studying second-order structure in learned optimizers and for designing future experiments that couple such structure to dynamic capacity and richer task curricula.

AI-Assisted Workflow and Author Contributions

This project was conducted using an AI-assisted research workflow in which large language model (LLM) agents were integrated into every stage of the process. Rather than focusing on any single narrow task (e.g., writing code or tuning hyperparameters), the author's role was to generate and refine the core conceptual ideas, decide which questions were worth testing experimentally, and steer the overall direction of the work, while treating the AI system as a high-bandwidth assistant for turning those ideas into concrete implementations.

Practically, this meant starting from an abstract hypothesis about the geometry of learning rules—that meta-trained optimizers might converge to a low-dimensional manifold rather than a single attractor—and using LLMs to help crystallize that hypothesis into specific experimental designs, model choices, and analysis procedures. Because the author is a beginner in this area of machine learning, the workflow deliberately assumed that the AI agent would often be more knowledgeable about existing techniques, implementation patterns, and standard analysis tools. The author's main contribution was therefore not to "out-code" the AI, but to (i) originate and iterate on novel abstractions and questions, (ii) remain open to the full "rule set" of machine learning methods surfaced by the AI, and (iii) selectively accept, modify, or reject AI-suggested directions based on whether they served the underlying research intent.

In practice, the LLM proposed candidate architectures, training loops, and diagnostic metrics; the author evaluated which of these aligned with the manifold/attractor hypothesis, requested variations when needed, and decided when to lock in a particular design and run experiments. All reported experiments were actually executed, and key numerical results (such as distance statistics and PCA dimensionality) were inspected and sanity-checked by the author. The conceptual framing of the work—in terms of second-order structure, optimizer manifolds, and links to self-improving systems—as well as the decision to treat this project as a case study in AI-accelerated science, originated from the author and was iteratively sharpened through this human–AI collaboration.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016.

[2] Luke Metz, Niru Maheswaranathan, C. Daniel Freeman, and Jascha Sohl-Dickstein. Learned optimizers scale and generalize. In *International Conference on Machine Learning*, 2019.

[3] Luke Metz, C. Daniel Freeman, Samuel S. Schoenholz, and Tal Kachman. Tasks, stability, architecture, and compute: Training more effective learned optimizers. In *Advances in Neural Information Processing Systems*, 2022.

[4] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gomez, and Nando de Freitas. Learned optimizers that scale and generalize. In *International Conference on Machine Learning*, 2017.

[5] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, 2018.

[6] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, 2018.

[7] Jürgen Schmidhuber. Evolutionary principles in self-referential learning. Diploma thesis, Institut für Informatik, Technische Universität München, 1987.

[8] Sebastian Thrun and Lorien Pratt, editors. *Learning to Learn.* Springer, 1998.