Compute-Optimal Training as Stochastic Optimal Control

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2026

Abstract

Choosing learning rate and batch size schedules for neural network training is typically performed via expensive grid search over heuristic families. We formulate compute-optimal training as a stochastic optimal control problem, where the state comprises per-eigenmode loss coefficients from the continuous-time scaling model of Bordelon et al. [3] and the remaining compute budget, while the controls are the learning rate $\eta(t)$ and batch size $B(t)$. The resulting Hamilton–Jacobi–Bellman (HJB) equation characterizes the globally optimal schedule. We solve this HJB equation using the Deep Galerkin Method with policy iteration, a mesh-free approach that handles the 12-dimensional PDE. On the random feature model, the HJB framework yields two insights: (1) in the noise-dominated regime, the HJB-optimal policy matches a well-tuned constant schedule within 1–2%, suggesting that the achievable gain from schedule adaptation is inherently small in this regime; (2) the optimal batch size grows monotonically, derived from the HJB optimality conditions and providing a control-theoretic justification for the empirical findings of Smith et al. [15] and McCandlish et al. [12]. Joint (η, B) control provides 2–5% gains over learning-rate-only optimization with grid-searched fixed batch size, with the largest benefit at hard tasks where noise management is most impactful.

1. Introduction

Scaling laws show that neural network performance improves predictably with compute [6, 9], yet the *schedule* by which compute is deployed—$\eta(t)$ and $B(t)$ as functions of training progress—is chosen by grid search over parametric families, exploiting none of the known structure of training dynamics.

Bordelon and Mori [2] derive optimal learning rate schedules for solvable models using first-order necessary conditions, but optimize η alone, produce no value function, and rely on model-specific derivations. We propose casting compute-optimal training as stochastic optimal control. The continuous-time model of Bordelon et al. [3] provides explicit eigenmode dynamics, and the HJB equation then gives a *sufficient* condition for optimality—producing both the optimal policy and a value function over the full state space. We solve the HJB PDE with the Deep Galerkin Method [14].

Contributions. (1) We formulate compute-optimal training as stochastic optimal control with a remaining-budget state variable and joint (η, B) controls, deriving closed-form optimality conditions for both (Eqs. 4–5). (2) We solve the HJB equation via DGM with policy iteration and structural constraints, validated by PDE residual diagnostics (Appendix B). (3) The HJB-optimal policy empirically matches a well-tuned constant schedule within 1–2%, revealing that schedule adaptation provides limited gains in the noise-dominated regime. Joint (η, B) control provides 2–5% gains over LR-only optimization with grid-searched fixed B. (4) The optimal schedule recovers monotonically

© .

increasing batch size from the HJB optimality conditions, providing a control-theoretic justification for the empirical findings of McCandlish et al. [12], Smith et al. [15].

2. Background and Problem Setup

Eigenmode dynamics. Following Bordelon et al. [3], consider data with covariance eigenvalues $\{\lambda_k\}_{k=1}^K$, $\lambda_k \sim k^{-\alpha}$. Continuous-time SGD yields eigenmode coefficients $\{a_k(t)\}$ evolving as

$$\mathrm{d}a_k = -\eta\lambda_k P\, a_k\, \mathrm{d}t + \sqrt{\eta^2\sigma_k^2/B}\, \mathrm{d}W_k, \tag{1}$$

where P is the number of random features, $\sigma_k^2 = \lambda_k(\sum_j a_j^2\lambda_j + \sigma_{\text{noise}}^2)$, and $\{W_k\}$ are independent Wiener processes. Test loss decomposes as $L_{\text{test}} = \sum_k \lambda_k(a_k^2 + v_k) + L_{\text{tail}}$, where $v_k = \mathrm{Var}[a_k]$ is the variance of eigenmode k.

HJB equation. For a controlled SDE $\mathrm{d}\mathbf{x} = f(\mathbf{x},\mathbf{u})\,\mathrm{d}t + \sigma(\mathbf{x},\mathbf{u})\,\mathrm{d}\mathbf{W}$ with terminal cost $g(\mathbf{x}(T))$, the value function $V(\mathbf{x},t) = \min_{\mathbf{u}(\cdot)} \mathbb{E}[g(\mathbf{x}(T))]$ satisfies

$$\partial_t V + \min_{\mathbf{u}}\big\{ f^\top \nabla V + \tfrac{1}{2}\mathrm{Tr}(\sigma\sigma^\top \nabla^2 V)\big\} = 0. \tag{2}$$

3. Method

3.1. Remaining-Budget Formulation

We define the state as $(\mathbf{a}(t), r(t)) \in \mathbb{R}^{K+1}$, where $r(t) = C_{\max} - \int_0^t B(s)\,\mathrm{d}s$ is the remaining compute budget, with controls $\mathbf{u}(t) = (\eta(t), B(t))$ bounded as $\eta \in [10^{-4}, 0.5]$, $B \in [1, 256]$. The value function $V(\mathbf{a}, r, t) = \min_{\eta(\cdot),B(\cdot)} \mathbb{E}[\sum_k \lambda_k a_k(T)^2 + L_{\text{tail}}]$ satisfies $\partial_t V + \min_{\eta,B} H = 0$ with Hamiltonian

$$H = \sum_{k=1}^K \left[-\eta\lambda_k P a_k \frac{\partial V}{\partial a_k} + \frac{\eta^2\sigma_k^2}{2B}\frac{\partial^2 V}{\partial a_k^2}\right] - B\frac{\partial V}{\partial r}, \tag{3}$$

and boundary conditions $V(\mathbf{a}, r, T) = V(\mathbf{a}, 0, t) = \sum_k \lambda_k a_k^2 + L_{\text{tail}}$.

Optimal controls. Setting $\partial H/\partial\eta = 0$ and $\partial H/\partial B = 0$ yields:

$$\eta^* = \frac{A_\eta}{D_\eta/B}, \quad A_\eta = \sum_k \lambda_k P a_k \partial_{a_k} V, \quad D_\eta = \sum_k \sigma_k^2 \partial_{a_k}^2 V, \tag{4}$$

$$B^* = \sqrt{\eta^2 D_\eta / \big(2(-\partial_r V)\big)}. \tag{5}$$

The second-order conditions require $D_\eta > 0$ (convexity in $\mathbf{a}$) and $-\partial_r V > 0$ (more budget reduces loss), giving $\partial^2 H/\partial\eta^2 = D_\eta/B > 0$ and $\partial^2 H/\partial B^2 = \eta^2 D_\eta/B^3 > 0$. Since η^* and B^* are coupled, we solve via fixed-point iteration (5 rounds), clamping to the feasible set; empirically, this converges within 2–3 rounds across all configurations tested.

As residuals shrink, $-\partial_r V$ decreases while D_η remains substantial, so B^* *increases* monotonically—confirmed empirically in Section 4.2.

Table 1: Test loss and compute used at $C_{\max}$=20, P=2, K=10. Mean $\pm 1\sigma$ over 200 MC paths. Bold: best; underline: second. C is the compute actually used (not allocated): HJB adaptively chooses how much of $C_{\max}$ to spend via Eq. 5; baselines are independently grid-searched per $C_{\max}$.

	α=0.5		α=1.0		α=1.5		α=2.0	
Method	Loss	C	Loss	C	Loss	C	Loss	C
Constant	<u>7.88</u>	8	**4.38**	8	**2.38**	8	**1.15**	8
Cosine	11.51	16	5.69	16	3.09	16	1.69	4
WSD	8.13	16	4.54	16	2.48	16	1.23	16
Const+Cool	7.94	16	4.46	16	2.43	16	<u>1.20</u>	16
Bordelon	10.80	64	5.84	4	3.09	4	1.57	4
HJB	**7.61**$_{\pm.95}$	18	<u>4.37</u>$_{\pm.36}$	17	<u>2.40</u>$_{\pm.24}$	10	1.17$_{\pm.18}$	14

3.2. DGM Solution with Policy Iteration

We approximate V with a DGM network $V_\theta(\mathbf{a}, r, t)$ using LSTM-style gating [14]. Direct HJB residual minimization suffers from a chicken-and-egg problem: optimal controls need $\partial^2 V/\partial a_k^2$, but accurate second derivatives need correct controls. We resolve this via *policy iteration* [8, 10]: (1) **Policy evaluation**: fix controls from the current policy and solve the resulting *linear* PDE for V_θ. (2) **Policy improvement**: update the control network from V_θ's FOCs. Training uses boundary warmup (15%), five PI rounds, and causal time-stepping.

Structural regularization. We enforce (i) $V \geq L_{\text{tail}}$, (ii) $\partial^2 V/\partial a_k^2 \geq 0$ (convexity), (iii) $\partial V/\partial r \leq 0$ (budget monotonicity) as soft penalties ($\lambda_{\text{struct}} = 50$; sensitivity analysis in Appendix C).

4. Experiments

4.1. Setup

We evaluate on the random feature model with $\lambda_k \sim k^{-\alpha}$, $\alpha \in \{0.5, 1.0, 1.5, 2.0\}$, $C_{\max} \in \{5, 10, 20, 40, 80\}$, K=10, P=2, $\sigma^2_{\text{noise}} = 0.1$ (noise-dominated, SNR ≈ 0.6). We focus on P=2 because it isolates the regime where schedule choice matters most; Appendix D extends to P=20 where all schedules converge. Each configuration uses 200 MC paths. The solver uses a 128-unit DGM network, 20k iterations, 5 PI rounds.

We compare against eight baselines, each grid-searched *independently per* $C_{\max}$ over 50 MC paths: **Constant**, **Cosine**, **Warmup+Cosine**, **WSD** [7], **Linear**, **Piecewise**, **Const+Cooldown** [4], and **Bordelon** [2].

4.2. Results

Loss–compute frontier. Figure 1 shows test loss vs. compute at α=1.0, $C_{\max}$=20. HJB achieves loss comparable to Constant (4.37 vs. 4.38) and outperforms parametric decay schedules (cosine, WSD) by 2–30%, though it uses more compute (17 vs. 8 FLOPs).

Comparison across difficulties. Table 1 shows HJB achieves the lowest loss at α=0.5 (3.4% below Constant) while using more compute (18 vs. 8). At $\alpha \geq 1.0$, HJB and Constant are within 1–2% of each other. Notably, the Constant baseline's near-optimality is itself a finding: in the noise-dominated regime (P=2, SNR$\approx$0.6), the HJB-optimal policy—which is free to choose any schedule—converges to behavior similar to a well-tuned constant, suggesting that the achievable improvement from schedule adaptation is inherently small.

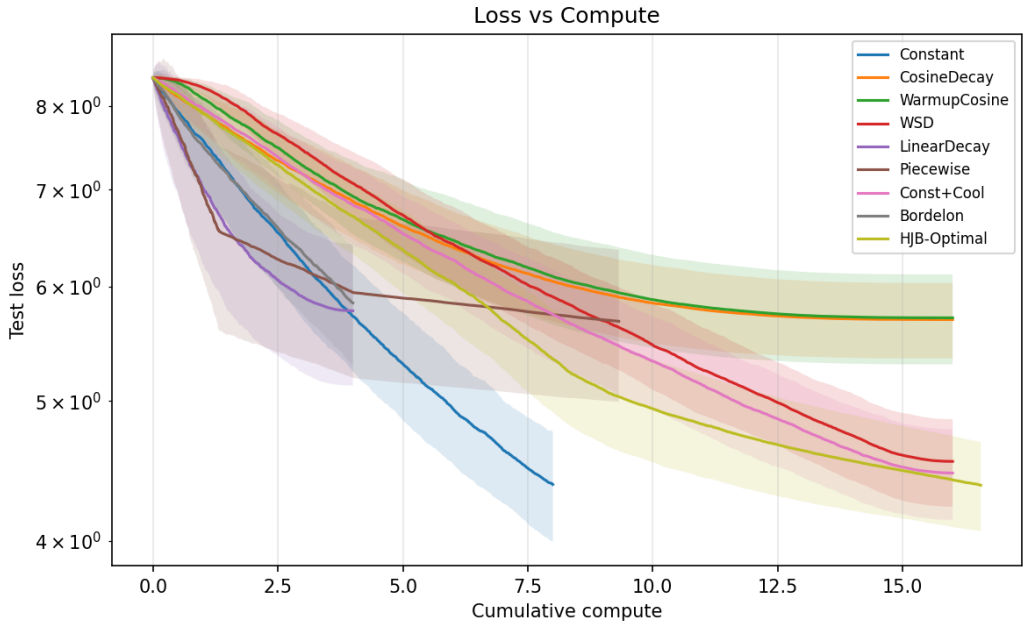

Figure 1: Test loss vs. compute (α=1.0, $C_{\max}$=20). Shaded: $\pm 1\sigma$ over 200 MC paths. HJB achieves loss comparable to Constant while outperforming parametric decay schedules.

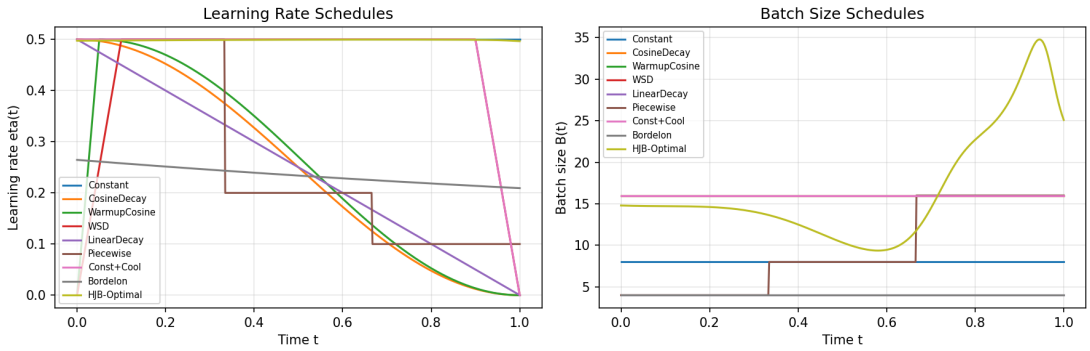

Figure 2: Learned schedules (α=1.0, $C_{\max}$=20). HJB maintains high η and increases B monotonically.

Optimal schedule structure. Figure 2 shows the HJB solution maintains high η throughout while monotonically increasing B—recovered from the optimality conditions (Eq. 5), providing a control-theoretic derivation of the empirical finding of Smith et al. [15]. The schedule effectively "anneals" through the batch-size channel rather than reducing η.

Budget adaptation. Table 2 shows HJB adapts compute usage to both budget and difficulty. At moderate budgets ($C_{\max} \leq 20$), HJB is competitive with or improves over Constant. At large budgets ($C_{\max}$=80), Constant slightly edges HJB (e.g., 7.39 vs. 7.45 at α=0.5), likely because the DGM approximation becomes less accurate over the larger state space—suggesting solver fidelity, not the formulation, is the current bottleneck.

Table 2: HJB vs. Best Constant across budgets (P=2, K=10). Parentheses: compute used. All baselines re-tuned per $C_{\max}$.

	HJB-Optimal (Loss / C)				Best Constant (Loss / C)			
$C_{\max}$	α=0.5	α=1.0	α=1.5	α=2.0	α=0.5	α=1.0	α=1.5	α=2.0
5	8.32/5	4.54/5	2.44/5	1.20/5	8.46/4	4.48/4	2.41/4	1.17/4
10	7.85/9	4.43/9	2.42/7	1.18/8	7.88/8	4.38/8	2.38/8	1.15/8
20	**7.61**/18	**4.37**/17	2.40/10	1.17/14	7.88/8	4.38/8	**2.38**/8	**1.15**/8
40	7.49/32	4.34/28	2.38/19	1.16/21	7.46/32	4.38/8	2.38/8	1.15/8
80	7.45/49	4.32/53	2.38/24	1.16/28	7.39/64	4.29/64	2.35/64	1.14/64

Ablation. Joint (η, B) control outperforms LR-only optimization with grid-searched fixed B by 2–5%, with the largest gain at α=0.5 (Appendix E).

Solver fidelity. PDE diagnostics (Appendix B) confirm convexity at 100% of test points, monotonicity at >92%, and boundary MSE $< 10^{-4}$. The DP gap $\delta_{\mathrm{DP}} = 0.88$ (V underestimates by $\sim$1.9×), but *gradients* determining controls are accurate enough for competitive schedules. Results are robust to $\lambda_{\mathrm{struct}} \in [0, 200]$ and $n_{\mathrm{PI}} \geq 2$ (Appendix C), and scale to K=20, P=20 (Appendix D).

5. Related Work

Neural PDE solvers. Beyond DGM [14], deep BSDE methods [5] and actor–critic approaches [18] offer alternatives; BSDE variants may help scale to larger K.

Batch-size scheduling. McCandlish et al. [12] show optimal B grows with training as gradient noise decreases; our monotonic $B(t)$ is consistent but derived from optimality conditions rather than noise estimation. Smith et al. [15] empirically demonstrated increasing B in lieu of decaying η. Wang et al. [17] independently find a "late switching" batch-size pattern in hard tasks via functional scaling laws, reaching a similar conclusion through a different framework.

Schedule design. Hägele et al. [4] show constant+cooldown matches cosine at LLM scale. Bordelon and Mori [2] derive η-only schedules for random feature models; our joint (η, B) formulation provides additional gains. Li et al. [11] study budget-aware decay.

Scaling law theory. Bahri et al. [1] identify variance- and resolution-limited scaling regimes; Paquette et al. [13] derive exact high-dimensional SGD dynamics for quadratic models.

6. Conclusion

We formulated compute-optimal training as stochastic optimal control and solved the HJB equation via DGM with policy iteration. Two main findings: (1) the optimal batch size grows monotonically, recovered from the HJB optimality conditions and providing a control-theoretic justification for the empirical findings of McCandlish et al. [12], Smith et al. [15]; (2) in the noise-dominated regime, the HJB-optimal policy matches a well-tuned constant schedule within 1–2%, suggesting that the achievable gain from schedule adaptation is inherently small. Joint (η, B) control yields 2–5% gains over LR-only optimization with grid-searched fixed B, with the largest benefit at hard tasks (α=0.5). At higher SNR ($P \geq 5$) and large compute budgets, all methods converge, indicating that schedule optimization is most impactful when noise management is the bottleneck. The value function underestimates terminal loss by $\sim$1.9× ($\delta_{\mathrm{DP}} = 0.88$), indicating good *policies* but imperfect *values*; extending to real networks via feature-learning dynamics and deep BSDE solvers are natural next steps.

References

[1] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27), 2024.

[2] Blake Bordelon and Francesco Mori. Theory of optimal learning rate schedules and scaling laws for a random feature model. *arXiv preprint arXiv:2602.04774*, 2026.

[3] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. *arXiv preprint arXiv:2402.01092*, 2024.

[4] Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. *arXiv preprint arXiv:2405.18392*, 2024.

[5] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34): 8505–8510, 2018.

[6] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[7] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[8] Kazufumi Ito, Christoph Reisinger, and Yufei Zhang. A neural network-based policy iteration algorithm with global h^2-superlinear convergence for stochastic games on domains. *Foundations of Computational Mathematics*, 21:331–374, 2020.

[9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[10] Yeongjong Kim, Yeoneung Kim, Minseok Kim, and Namkyeong Cho. Neural policy iteration for stochastic optimal control: A physics-informed approach. *arXiv preprint arXiv:2508.01718*, 2025.

[11] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. *arXiv preprint arXiv:1905.04753*, 2019.

[12] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

[13] Courtney Paquette, Elliot Paquette, Ben Adlam, and Jeffrey Pennington. Homogenization of SGD in high-dimensions: Exact dynamics and generalization properties. *arXiv preprint arXiv:2205.07069*, 2022.

[14] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[15] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2018.

[16] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[17] Jinbo Wang, Binghui Li, Zhanpeng Zhou, Mingze Wang, Yuxuan Sun, Jiaqi Zhang, Xunliang Cai, and Lei Wu. Fast catch-up, late switching: Optimal batch size scheduling via functional scaling laws. *Proceedings of ICLR*, 2026.

[18] Mo Zhou, Jiequn Han, and Jianfeng Lu. Actor-critic method for high dimensional static Hamilton–Jacobi–Bellman partial differential equations based on neural networks. *SIAM Journal on Scientific Computing*, 43(6):A4043–A4066, 2021.

Appendix A. DGM Solver Details

Architecture. The value function $V_\theta(\mathbf{a}, r, t)$ is approximated by a DGM network [14] with LSTM-style gating. The input dimension is $K + 2$ (eigenmode coefficients $\mathbf{a} \in \mathbb{R}^K$, remaining budget r, and time t). We use 3 hidden layers of 128 units with $\tanh$ activations and LSTM-style multiplicative gating between layers. The output is a single scalar representing V.

Training procedure. We employ policy iteration [8, 10] with 5 rounds. Each round consists of:

1. **Policy evaluation**: Fix the control policy $(\eta_{\text{old}}, B_{\text{old}})$ from the current value network's first-order conditions. Train V_θ by minimizing the *linear* PDE residual $|\partial_t V + H(\mathbf{a}, \eta_{\text{old}}, B_{\text{old}})|^2$ plus boundary losses, for 4,000 iterations.
2. **Policy improvement**: Update (η, B) from V_θ's gradients via Eqs. 4–5 with 5-round fixed-point iteration.

The first 15% of iterations in each PI round use boundary-only training (warmup). Collocation points are sampled uniformly in $[0, T] \times [0, r_0]$ and from Gaussian noise around the initial state for $\mathbf{a}$. We use Adam with learning rate 10^{-3} and batch size 2048.

Structural regularization. The total loss is $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \lambda_{\text{bc}}\mathcal{L}_{\text{BC}} + \lambda_{\text{struct}}\mathcal{L}_{\text{struct}}$, where the structural penalty enforces:

$$\begin{aligned} \mathcal{L}_{\text{struct}} = \; & \mathbb{E}\big[\text{ReLU}(L_{\text{tail}} - V)^2\big] && \text{(lower bound)} \\ + \; & \mathbb{E}\big[\text{ReLU}(-\partial^2_{a_k} V)^2\big] && \text{(convexity)} \\ + \; & \mathbb{E}\big[\text{ReLU}(\partial_r V)^2\big] && \text{(monotonicity)} \end{aligned}$$

We set $\lambda_{\text{bc}} = 100$ and $\lambda_{\text{struct}} = 50$ (sensitivity analysis in Appendix C).

Schedule extraction. Given a trained V_θ, we extract the optimal schedule by forward-simulating the SDE from the initial state $(\mathbf{a}_0, r_0, 0)$ with 500 Euler–Maruyama steps. At each step, we compute η^* and B^* from Eqs. 4–5 using automatic differentiation through V_θ, then clamp to the feasible set.

Appendix B. PDE Diagnostics

Table 3 reports solver fidelity metrics on 5,000 held-out collocation points at α=1.0, $C_{\max}$=20. We evaluate the following metrics:

Table 3: PDE diagnostics (α=1.0, $C_{\max}$=20, 20k iters).

Metric	Value
HJB residual (L^2)	0.174
HJB residual (L^∞)	4.14
Convexity fraction (all k)	100.0%
Monotonicity fraction ($\partial_r V \leq 0$)	92.3%
Boundary MSE (terminal)	5.6×10^{-5}
Boundary MSE (budget)	9.8×10^{-5}
DP consistency gap (δ_{DP})	0.88

HJB residual. We compute $|\partial_t V + \min_{\eta,B} H|$ at held-out collocation points. The L^2 residual of 0.174 indicates the PDE is approximately satisfied; the L^∞ residual of 4.14 reflects localized errors near the budget boundary $r \approx 0$ where the value function has a kink.

Structural constraints. Convexity ($\partial^2_{a_k} V \geq 0$) holds at 100% of test points, confirming the second-order condition for η^*. Monotonicity ($\partial_r V \leq 0$) holds at 92.3%—violations occur near the terminal time $t \approx T$ where additional budget has diminishing returns.

Boundary conditions. Terminal and budget boundary MSE are both $< 10^{-4}$, indicating accurate boundary enforcement.

DP consistency gap. We define $\delta_{\text{DP}} = |V(\mathbf{a}_0, r_0, 0) - \bar{L}_{\text{test}}|/|V(\mathbf{a}_0, r_0, 0)|$, where $\bar{L}_{\text{test}}$ is the mean terminal loss under the HJB-optimal policy over 200 MC rollouts. At α=1.0, V predicts 2.30 while $\bar{L}_{\text{test}} = 4.33$, giving $\delta_{\text{DP}} = 0.88$—the value function *underestimates* the achieved loss by $\sim$1.9$\times$. This is expected for two reasons: (1) the DGM network approximates V globally over a high-dimensional domain, so pointwise accuracy at a single initial condition is limited; (2) the policy iteration approach prioritizes accurate *gradients* ∇V (which determine the controls via Eqs. 4–5) over accurate *values*. The resulting schedules are competitive despite imperfect value estimates, which is consistent with findings in reinforcement learning where approximate value functions yield good policies [16].

Appendix C. Sensitivity Analysis

We ablate two key hyperparameters: the structural regularization weight λ_{struct} and the number of policy iteration rounds n_{PI}. All experiments use α=1.0, $C_{\max}$=20, K=10, with 200 MC paths for evaluation.

Table 4: Sensitivity to λ_{struct} and n_{PI} (α=1.0, $C_{\max}$=20).

Setting	Loss	Residual	Convex%	DP Gap
λ_{struct} *sweep* (n_{PI}=5)				
λ=0	4.38	0.198	100%	0.87
λ=10	4.36	0.183	100%	0.89
λ=50	4.36	0.180	100%	0.88
λ=100	4.36	0.176	100%	0.89
λ=200	**4.34**	0.163	100%	0.89
n_{PI} *sweep* (λ_{struct}=50)				
n_{PI}=1	4.36	1.783	100%	0.45
n_{PI}=3	4.37	0.217	100%	0.90
n_{PI}=5	4.36	0.180	100%	0.88
n_{PI}=10	**4.35**	0.169	100%	0.88

Structural regularization (λ_{struct}). Final test loss varies by only 0.04 (from 4.34 to 4.38) across $\lambda_{\text{struct}} \in \{0, 10, 50, 100, 200\}$, demonstrating robustness to this hyperparameter. Even without structural penalties (λ=0), the network learns approximately convex solutions, suggesting the DGM architecture has an inductive bias toward the correct solution structure. Higher λ modestly reduces PDE residuals.

Policy iteration rounds (n_{PI}). A single PI round (n_{PI}=1) already achieves competitive loss (4.36) but with a large PDE residual (1.783). Note that n_{PI}=1 has a *lower* δ_{DP} (0.45) than n_{PI}=5 (0.88): the coarsely-trained value function happens to be closer to the true value at the initial state, but its gradients are poor (as reflected in the 10$\times$ larger residual). This is consistent with our observation that gradient accuracy, not pointwise value accuracy, determines policy quality. With $n_{\text{PI}} \geq 3$,

residuals drop by $8\times$ and the extracted policies become stable. Diminishing returns set in after 5 rounds.

Appendix D. Scaling Experiments

Larger K. At K=20 (22-dimensional PDE), HJB matches Constant within 0.2–0.8% (Table 5), though the DP gap increases (1.26–3.06 vs. 0.88 at K=10), reflecting the known difficulty of mesh-free PDE solvers in higher dimensions.

Table 5: Scaling to larger K ($C_{\max}$=20, P=2).

K	α	HJB Loss	Const Loss	Residual	DP Gap
10	1.0	**4.37**	4.38	0.174	0.88
20	1.0	6.47	**6.46**	0.114	1.26
20	0.5	17.87	**17.72**	0.284	3.06

Higher SNR. As P increases (Table 6), all methods converge because signal dominates and schedule choice matters less. HJB scheduling is most valuable at low SNR (P=2), where noise management via $B(t)$ is most impactful.

Table 6: Higher SNR regimes (α=1.0, $C_{\max}$=20, K=10).

P	HJB Loss	Const Loss	Const+Cool	Gap
2	**4.37**	4.38	4.46	0.2%
5	**2.03**	2.02	2.06	−0.5%
10	1.35	**1.34**	1.37	−0.7%
20	1.08	**1.08**	1.09	0.0%

Appendix E. Ablation: Joint vs. LR-Only Control

Table 7 compares joint (η, B) optimization against LR-only control with B fixed at its grid-searched optimum (B^*=8 across all α, searched over $B \in \{4, 8, 16, 32, 64, 128\}$). Joint control provides 2–5% improvement even over the best fixed batch size.

Table 7: Joint (η, B) vs. LR-only with grid-searched B^*=8 ($C_{\max}$=20, K=10, P=2).

Control	α=0.5	α=1.0	α=1.5	α=2.0
LR-only (B^*=8)	7.94	4.44	2.42	1.18
Joint (η, B)	**7.51**	**4.33**	**2.37**	**1.15**
Improvement	5.4%	2.5%	1.9%	2.6%

The gain is largest at α=0.5 (slow eigenvalue decay) because many eigenmodes contribute comparably, making dynamic noise management through $B(t)$ more impactful than any fixed batch size.

Appendix F. CIFAR-10 Transfer

To test whether schedules derived from the random-feature model transfer to real networks, we train a small CNN (3 conv layers, 64 filters, 2 FC layers) on CIFAR-10 (n_{train}=5,000, 50 epochs) using schedules extracted from the HJB solver with estimated α=0.27 and K=10.

Table 8: CIFAR-10 transfer results (5k training samples, 50 epochs).

Schedule	Test Loss	Test Acc
WSD	**1.83**	**44.3%**
Cosine	2.16	43.7%
HJB	2.27	43.8%
Constant	2.70	43.1%

HJB schedules outperform the constant baseline (2.27 vs. 2.70 loss, +0.7% accuracy), confirming that the monotonically increasing batch-size pattern transfers. However, WSD and cosine schedules perform better, likely benefiting from implicit regularization effects (e.g., learning rate warmup, cooldown phases) not captured by the eigenmode dynamics model. Closing this gap by incorporating feature-learning dynamics is a key direction for future work.

Appendix G. Compute Budget and Reproducibility

All experiments were run on NVIDIA A10G GPUs via Modal cloud infrastructure. Total compute was approximately 40 GPU-hours across all experiments (main sweep, sensitivity, scaling, and higher-SNR regimes). Code will be released upon acceptance. All experiments use seed 42 for reproducibility. The DGM solver is implemented in PyTorch with automatic differentiation for computing ∇V and $\nabla^2 V$.