Stability-Guided Meta-Control for Unsupervised Bilevel Meta-Learning

Liz Lemma Future Detective

January 20, 2026

Abstract

Unsupervised meta-learning methods such as DHM-UHT place task construction (pseudo-labeling via clustering) inside the inner loop, enabling robustness to label noise and heterogeneous tasks—but at substantial compute cost, especially when using MAML-style inner updates and expensive clustering. Inspired by the source material's representation-stability analysis (SVCCA), we propose turning stability from a diagnostic into a control signal. We formalize training as an online decision problem: at each outer iteration, the learner chooses among discrete inner-loop strategies (e.g., ANIL vs MAML, number of steps) and task-construction granularities (e.g., clustering resolution/outlier thresholds), trading off accuracy and compute. We introduce cheap stability proxies based on CKA over random projections and EMA checkpoints, and design a stability-guided cost-sensitive (contextual) bandit controller. Theoretically, we prove $\tilde{O}(\sqrt{TK})$ regret ($K = |\mathcal{A}|$) for the compute-regularized objective with a matching lower bound, and show that if stability predicts when deeper inner updates are beneficial, the controller approaches an oracle that always chooses the best compute level. Empirically (implementation recommended), we evaluate on the DHM-UHT benchmark suite and larger backbones to demonstrate self-tuning behavior: ANIL-like compute in easy/stable regimes and MAML-like adaptation only when instability indicates drifting pseudo-labels or heterogeneous task difficulty.

Table of Contents

3. 3. Problem Formulation: online selection of inner-loop strategy and constructor granularity; define action space, cost, stability signal, and compute–accuracy objective; discuss evaluation metrics (accuracy vs wall-clock vs gradient steps).

4. 4. Stability Signals: define CKA/SVCCA proxies that are cheap; EMA checkpoints; random projection CKA; probe batch selection; invariances/limitations; calibration diagnostics.

5. 5. Algorithm: Stability-Guided Cost-Sensitive Bandit (and contextual variant): exploration, cost-aware updates, stability-conditioned action filtering; practical variants (thresholding, two-stage gating).

6. 6. Main Theorems: regret bounds vs best fixed action and piecewise-stationary comparator; stability-informativeness implies near-oracle compute selection; matching lower bound from adversarial bandits.

7. 7. Complexity and Optimality: time/space overhead, dependence on probe batch size, number of actions; discussion of what is information-theoretically unavoidable; sensitivity to stability noise.

8. 8. Implementation & Experimental Plan (recommended): reproduce DHM-UHT settings; extend to larger backbones; controlled regimes (label noise, heterogeneity, cluster drift); ablations (stability-only vs loss-only vs random).

9. 9. Limitations and Extensions: continuous action spaces (tuning eps/min_samples), contextual policies learned by small networks, multi-objective constraints, and transfer of controllers across datasets.

10. 10. Conclusion: stability-guided self-tuning meta-learning; implications for 2026-scale unsupervised adaptation.

1 Introduction

Unsupervised bilevel meta-learning has recently emerged as a practical mechanism for learning transferable representations without annotated tasks by *constructing* episodic supervision on the fly. In this setting, an outer loop updates a shared backbone network using a meta-objective, while an inner loop adapts a task-specific module using pseudo-labels or partitions produced from the current minibatch by a task constructor (e.g., clustering in representation space). Such pipelines inherit much of the algorithmic structure of supervised meta-learning, but the absence of ground-truth labels introduces an additional degree of freedom: the task itself is a stochastic function of the model parameters and of the chosen construction granularity. Consequently, training dynamics depend not only on the underlying optimizer, but also on how aggressively we adapt per batch and how finely we partition data into pseudo-classes.

A persistent obstacle to scaling these methods is that the most accurate variants often require expensive inner-loop computation, frequent reconstruction of pseudo-tasks, or both. In particular, deep inner adaptation (e.g., updating the backbone during inner-loop steps) and high-resolution task construction (e.g., more clusters, additional refinement passes) can substantially increase wall-clock time per outer iteration. Conversely, lightweight updates (e.g., adapting only a head) can be markedly cheaper, but may underperform when the induced pseudo-tasks drift rapidly or when the representation is not yet sufficiently stable for shallow adaptation to be reliable. This creates a compute–accuracy tension that is not naturally resolved by fixing a single algorithmic choice throughout meta-training: the *right* level of inner-loop effort is typically nonstationary, varying across phases of training and across regimes of data heterogeneity and pseudo-label noise.

We formalize this tension by introducing a finite set of *actions* $\mathcal{A}$, where each action encodes (i) an inner-loop update rule and its depth (e.g., ANIL-like versus MAML-like variants, number of inner steps), and (ii) a constructor granularity (e.g., clustering resolution or refinement schedule). Each action $a \in \mathcal{A}$ has an associated compute cost $c(a)$, which we treat as known or measurable, and induces a meta-loss $\ell_t(a)$ when executed at outer iteration t. The central difficulty is informational: under the usual online training protocol, we observe only the loss of the chosen action, i.e., bandit feedback, so we cannot directly compare all candidate strategies at each iteration. Thus, selecting compute adaptively cannot be reduced to an offline hyperparameter sweep, nor can it be solved by a standard full-information controller.

Our guiding observation is that, although meta-losses are unobserved for unchosen actions, we can often obtain cheap *side information* about the state of training. Specifically, we assume access to a representation stability signal $s_t \in [0, 1]$, computed with negligible overhead from a fixed probe batch via a similarity functional between consecutive checkpoints (e.g., CKA

or SVCCA-style proxies). Intuitively, high stability indicates that the backbone representation is changing slowly, so aggressive inner-loop updates and frequent high-resolution reconstruction may yield diminishing returns; low stability indicates substantial drift, suggesting that additional inner adaptation and/or more careful task construction may be needed to prevent mismatch between the induced pseudo-tasks and the evolving representation. The key point is that s_t can be computed independently of the action choice with cost $o \ll c(a)$, making it a candidate signal for compute allocation.

We therefore pose the following online decision problem: at each outer iteration t, we choose an action $a_t \in \mathcal{A}$ to execute one meta-update, with the aim of minimizing a compute-regularized objective

$$\sum_{t=1}^{T} \big(\ell_t(a_t) + \lambda c(a_t)\big),$$

where $\lambda \geq 0$ encodes a user-specified trade-off between accuracy and compute. This Lagrangian formulation also subsumes constrained variants in which one seeks to minimize $\sum_t \ell_t(a_t)$ subject to $\sum_t c(a_t) \leq B$. Importantly, the losses $\ell_t(a)$ may be nonstationary, since both the pseudo-tasks and the backbone parameters evolve over time; hence, the controller must operate robustly under potentially adversarial or drifting losses.

Our main contribution is a stability-guided, cost-sensitive bandit controller for this online bilevel training loop. At a high level, we combine two ideas. First, we employ an EXP3-style multiplicative-weights algorithm on the cost-regularized loss, which yields regret guarantees under bandit feedback without assuming stochasticity. Second, we optionally incorporate a *stability gate* that restricts exploration to subsets of actions deemed plausible given the current stability s_t; this heuristic is analytically accounted for by comparing against an appropriate comparator class or by bounding the number of rounds on which gating may exclude near-optimal actions. The resulting controller is lightweight—maintaining only $O(|\mathcal{A}|)$ weights—and can be inserted into existing unsupervised meta-learning code with minimal intrusion.

We complement the algorithm with theory that matches the informational structure of the problem. In the adversarial regime, we prove an expected regret bound of order $\tilde{O}(\sqrt{T|\mathcal{A}|})$ for the cost-regularized objective, up to the natural scaling factor induced by the range of $\ell_t(a) + \lambda c(a)$. We also provide a matching $\Omega(\sqrt{T|\mathcal{A}|})$ lower bound, demonstrating that one cannot generally improve the dependence on T and $|\mathcal{A}|$ under bandit feedback without additional assumptions. Since algorithmic preferences can shift over training, we further extend the analysis to piecewise-stationary comparators, yielding guarantees against a policy that changes its preferred action a bounded number of times.

Finally, we articulate an additional "stability informativeness" condition

under which the stability signal is predictive of the performance gap between cheap and expensive action families. Under this calibration assumption, a simple stability-based gating rule yields near-oracle compute selection: the controller pays only an additive $O(\epsilon T)$ penalty relative to a hypothetical policy that, at each iteration, could evaluate all actions and choose the best compute–accuracy trade-off. This result formalizes the intuition that stability can serve not merely as a diagnostic, but as actionable information for allocating inner-loop compute.

Empirically, we propose to evaluate the controller on unsupervised meta-learning benchmarks with in-the-loop task construction, varying backbone scale, data heterogeneity, and pseudo-label noise. The aim is to demonstrate that the controller attains the accuracy of compute-intensive inner updates when instability demands it, while reverting to ANIL-like costs in regimes where representations stabilize and expensive adaptation becomes redundant.

2 Background and Motivation

We briefly recall the unsupervised meta-learning loop with in-the-loop task construction that motivates our controller, using DHM-UHT as a representative instance. The salient feature is that, at each outer iteration, we must *manufacture* an episodic supervision signal from an unlabeled minibatch, and this manufactured task depends on the current representation. Consequently, both the quality of the pseudo-task and the value of inner-loop adaptation vary across training, and the appropriate compute expenditure is inherently nonstationary.

Unsupervised tasks from a minibatch. Let $T_t = \{x_i\}_{i=1}^n$ denote the minibatch sampled at outer iteration t. Given the current backbone f_{θ_t}, we form embeddings $z_i = f_{\theta_t}(x_i) \in \mathbb{R}^d$. A task constructor g maps the embeddings to a partition or pseudo-labeling, for instance by clustering:

$$\hat{y}_i \;=\; g\big(\{z_j\}_{j=1}^n; r\big)_i,$$

where r denotes a granularity parameter (e.g., number of clusters, refinement passes, temperature). From $\{(x_i, \hat{y}_i)\}$ we then sample an episode by splitting indices into a support set S_t and a query set Q_t, typically with an N-way K-shot structure induced by the pseudo-classes. The resulting "task" is thus a random object depending on both data and current parameters; in particular, changing θ_t changes the clustering geometry and can alter $\hat{y}_i$ discontinuously.

Inner-loop adaptation with a reinitialized head. Given a constructed task, we introduce a task-specific module h (e.g., a linear classifier, a small MLP, or an adapter) with parameters ϕ. In DHM-UHT-style procedures, ϕ

is (re)initialized per task, either from a fixed initialization ϕ_0 or from a data-dependent initialization (a "dynamic head") computed from the support set, e.g., by prototype statistics or a least-squares fit. We then perform k inner steps on the support loss

$$\phi^{(i+1)} \;=\; \phi^{(i)} - \alpha \nabla_{\phi} \mathcal{L}_{S_t}\big(f_{\theta_t}, h_{\phi^{(i)}}\big), \qquad i = 0, \ldots, k-1,$$

with $\phi^{(0)}$ given by the reinitialization rule. The algorithmic choice that matters for compute is whether we adapt only ϕ (ANIL-like) or also (part of) θ during the inner loop (MAML-like), and how many inner steps k are taken. These choices directly affect the number of forward/backward passes and, when combined with clustering resolution r, determine the per-iteration cost.

Outer-loop update and the role of pseudo-task drift. After inner adaptation, the outer update uses the query set Q_t to update the shared backbone:

$$\theta_{t+1} \;=\; \theta_t - \beta \nabla_{\theta} \mathcal{L}_{Q_t}\big(f_{\theta_t}, h_{\phi^{(k)}}\big),$$

where the gradient may be computed with full meta-gradients (backpropagating through inner steps) or with first-order approximations, depending on the action chosen. In unsupervised task construction, the query loss depends on pseudo-label quality; if g produces inconsistent partitions as θ_t evolves, then the effective objective is noisy and may require more aggressive inner adaptation or more conservative construction to avoid chasing transient pseudo-classes. Conversely, when the representation evolves slowly and partitions stabilize, the marginal utility of expensive inner updates can diminish.

Representation stability as cheap side information. We therefore seek a scalar signal indicating whether the representation is currently drifting. We assume access to a fixed probe batch $P = \{x_j^{\mathrm{p}}\}_{j=1}^{m}$, drawn from the same distribution as training data but not used for optimization. Let

$$F_t \;=\; \begin{bmatrix} f_{\theta_t}(x_1^{\mathrm{p}})^{\top} \\ \vdots \\ f_{\theta_t}(x_m^{\mathrm{p}})^{\top} \end{bmatrix} \in \mathbb{R}^{m \times d}$$

collect probe representations. A stability functional $S(\theta_t, \theta_{t-1}; P)$ measures similarity between F_t and F_{t-1}, yielding $s_t \in [0, 1]$ where larger values indicate greater stability. Two standard choices are SVCCA-style similarity and linear CKA. For instance, a common CKA proxy is

$$\mathrm{CKA}(F_t, F_{t-1}) \;=\; \frac{\big\|F_t^{\top} F_{t-1}\big\|_F^2}{\big\|F_t^{\top} F_t\big\|_F \,\big\|F_{t-1}^{\top} F_{t-1}\big\|_F}, \qquad s_t := \mathrm{CKA}(F_t, F_{t-1}),$$

possibly after centering rows and applying a random projection to reduce the effective dimension. The key property for our purposes is computational: since m is small and fixed, and since the computation is a pairwise comparison of probe features, the overhead o is negligible relative to any full bilevel update. Moreover, s_t is available regardless of which action we take at iteration t, and thus constitutes valid side information in an online controller.

Empirical motivation: stability separates regimes. Empirically, stability behaves differently across algorithmic choices and data regimes. First, deep inner-loop variants that modify the backbone (or use many inner steps) tend to exhibit larger representation drift early in training, reflected in smaller s_t, as the model rapidly reshapes its embedding space. Head-only adaptation typically yields higher s_t because the backbone changes only through the outer gradient. Second, task-constructor granularity interacts strongly with stability: increasing clustering resolution or adding refinement passes can increase the sensitivity of pseudo-labels to small representation changes, inducing oscillatory partitions and reducing s_t even when outer learning rates are unchanged. Third, exogenous factors such as batch heterogeneity (e.g., implicit domain shifts across minibatches) and pseudo-label noise can reduce stability by forcing the backbone to accommodate incompatible local clusterings; in such regimes, shallow adaptation can underfit, whereas deeper adaptation or more careful construction can improve the effective query loss.

These observations suggest that s_t can be used as a diagnostic for when additional compute is likely to be beneficial. In stable phases, expensive inner computation often yields diminishing returns relative to its cost, whereas in unstable phases it may be necessary to prevent mismatch between rapidly changing pseudo-tasks and the learner. The technical challenge is that we do not observe $\ell_t(a)$ for unchosen actions, so we cannot directly validate this heuristic online; hence we require a principled bandit-style mechanism that can exploit s_t without assuming full information. This motivates the formal online control problem and action-based compute accounting introduced next.

3 Problem Formulation

We formalize the control of inner-loop compute and task-constructor granularity as an online decision problem coupled to an unsupervised bilevel meta-learner. The salient difficulty is informational: at each outer iteration we may choose among multiple algorithmic variants, but we only observe the resulting query loss for the variant we actually run. Our aim is to select variants so as to trade off accuracy and compute in a principled way, while

permitting the use of a cheap stability signal as side information.

Outer iterations as rounds; tasks as random batches. Let D be an unlabeled dataset (or stream), and let $T \in \mathbb{N}$ denote the training horizon (number of outer-loop iterations). At round $t \in \{1, \ldots, T\}$ we sample a minibatch T_t from D and treat it as an implicit meta-task. For a fixed shared backbone f_{θ_t} at the start of the round, we may run one outer meta-update using any admissible combination of (i) how we construct pseudo-labels/partitions from T_t and (ii) how we adapt within the induced episode prior to the outer gradient step.

Action space encodes algorithmic variants. We assume a finite set of actions $\mathcal{A}$ with $|\mathcal{A}| = K$. Each $a \in \mathcal{A}$ specifies a complete "inner-loop strategy + constructor granularity" recipe for a single outer iteration. Concretely, an action may encode: (i) the task constructor setting (e.g., clustering resolution r, refinement passes, temperature), (ii) the inner-loop adaptation rule (e.g., head-only versus including part of the backbone, number of inner steps k, first-order versus higher-order meta-gradient), and (iii) any other discrete knobs that materially change compute. We denote by $\ell_t(a) \in [0, 1]$ the meta-loss incurred on the query set of the constructed episode *if* we were to execute round t using action a. The normalization $[0, 1]$ is without loss of generality and is enforced in practice by rescaling or clipping (e.g., by dividing by a fixed constant and truncating to $[0, 1]$).

Compute costs as known arm-dependent penalties. Each action has an associated compute cost $c(a) \in [1, C]$, assumed known (or pre-measured) up to a fixed scale, where C is the maximal per-round cost among actions. We treat $c(a)$ as an abstract compute budget unit, chosen to correlate with wall-clock time. Typical instantiations include a weighted sum of: the number of forward/backward passes through the backbone, the number of inner-loop gradient evaluations, and the number of task-constructor calls (e.g., clustering iterations). In this work we take $c(a)$ to be action-dependent but time-invariant; this matches the common situation in which each variant has a predictable per-iteration cost profile, even though the realized wall-clock time may vary mildly due to hardware effects.

Stability as cheap side information. At each round t we assume access to a scalar signal $s_t \in [0, 1]$ measuring representation stability between successive checkpoints, computed on a small fixed probe batch P. We model s_t as available *prior* to selecting a_t, with overhead o negligible relative to $\min_{a \in \mathcal{A}} c(a)$. We emphasize that s_t is not a loss and is not required to be predictive; it is merely auxiliary information that may be exploited by a controller to bias exploration or to gate expensive actions. The precise choice

of stability functional $S(\theta_t, \theta_{t-1}; P)$ is deferred to the next section.

Online control protocol and bandit feedback. A controller chooses an action $a_t \in \mathcal{A}$ at each round according to past observations and the current stability signal. The interaction is:

1. observe s_t (and any other admissible history-dependent state);
2. sample a_t (possibly randomly) and execute one outer iteration of the meta-learner under a_t, yielding θ_{t+1};
3. observe the incurred bandit feedback $\ell_t(a_t)$ and the cost $c(a_t)$.

Crucially, we do *not* observe $\ell_t(a)$ for any unchosen $a \neq a_t$. This is the standard bandit setting; our results do not rely on full-information access.

Compute–accuracy objective: Lagrangian and constrained forms. We study two equivalent formulations, depending on whether compute is treated as a penalty or a hard budget. In the Lagrangian form, for a fixed trade-off $\lambda \geq 0$, the controller seeks to minimize

$$\sum_{t=1}^{T} \Big(\ell_t(a_t) + \lambda \, c(a_t) \Big). \tag{1}$$

Define the cost-regularized per-round loss $L_t(a) := \ell_t(a) + \lambda c(a) \in [0, 1+\lambda C]$. The Lagrangian formulation is convenient analytically, since it reduces to an adversarial bandit problem over bounded losses $L_t(a)$.

Alternatively, in the constrained form, given a total compute budget B, we aim to minimize the unregularized query losses subject to $\sum_{t=1}^{T} c(a_t) \leq B$. Standard duality heuristics suggest that for appropriate λ the Lagrangian controller yields near-feasible and near-optimal solutions in the constrained sense; in our experiments we report both the achieved losses and the realized compute consumption to expose the trade-off directly.

Performance metric: regret against natural comparators. To quantify the online penalty for not knowing which action is best in advance, we measure regret relative to a comparator class. For the basic case of a single best fixed action in hindsight, we define

$$R_T := \mathbb{E}\Big[\sum_{t=1}^{T} L_t(a_t) \Big] - \min_{a \in \mathcal{A}} \sum_{t=1}^{T} L_t(a),$$

where the expectation is over the controller's internal randomness (and any algorithmic randomness in task construction and optimization). We also consider nonstationary comparators (piecewise-constant best actions), motivated by the empirically observed regime changes during training.

How we evaluate compute–accuracy trade-offs empirically. The formal objective (1) is a training-time proxy for what we ultimately care about: downstream generalization under a compute budget. Accordingly, we report (i) downstream evaluation accuracy after meta-training (e.g., few-shot accuracy on held-out episodes or linear evaluation), (ii) total wall-clock time, and (iii) normalized compute in gradient-evaluation units $\sum_{t=1}^{T} c(a_t)$. The last metric is the one directly controlled by our formulation; wall-clock time is reported to validate that $c(a)$ is a faithful surrogate. Finally, by sweeping λ (or imposing budgets B) we obtain a Pareto curve of accuracy versus compute, which allows direct comparison to fixed-action baselines (e.g., always-ANIL-like or always-MAML-like) and to stability-agnostic bandit controllers.

4 Stability Signals

We now specify the stability functional $S(\theta_t, \theta_{t-1}; P)$ used to form the side-information scalar $s_t \in [0, 1]$. Our design requirement is that s_t (i) be computable at negligible overhead relative to any admissible action, (ii) reflect nontrivial changes in the representation induced by the backbone update, and (iii) be sufficiently robust to nuisance transformations (e.g., feature rescaling) to admit consistent thresholding and calibration.

Probe representations and a generic stability template. Let $P = \{x_i\}_{i=1}^{n_P}$ be a fixed probe batch (or a small fixed collection of batches) drawn once and held constant. For a chosen probe layer $\varphi(\cdot; \theta) \in \mathbb{R}^d$ (typically the penultimate backbone features), define the probe feature matrices

$$Z_t \in \mathbb{R}^{n_P \times d}, \qquad (Z_t)_{i,:} := \varphi(x_i; \theta_t).$$

A large class of stability measures can be written as

$$s_t := S(\theta_t, \theta_{t-1}; P) := \mathcal{S}(Z_t, Z_{t-1}),$$

where $\mathcal{S}$ outputs a scalar in $[0, 1]$, with larger values indicating greater representational similarity between successive checkpoints. In practice we compute Z_t and Z_{t-1} without gradients and cache Z_{t-1} from the previous round, so that the marginal per-round overhead is one forward pass of the probe batch (plus a small amount of linear algebra).

Linear CKA as a cheap default. We take as our default $\mathcal{S}$ the linear centered kernel alignment (CKA), which we recall in a form convenient for implementation. Let $\tilde{Z}$ denote row-centered features, $\tilde{Z} := HZ$ with $H := I_{n_P} - \frac{1}{n_P}\mathbf{1}\mathbf{1}^\top$. The linear CKA similarity between Z and Z' is

$$\mathrm{CKA}(Z, Z') := \frac{\|\tilde{Z}^\top \tilde{Z}'\|_F^2}{\|\tilde{Z}^\top \tilde{Z}\|_F \, \|\tilde{Z}'^\top \tilde{Z}'\|_F} \in [0, 1]. \tag{2}$$

We then set $s_t := \mathrm{CKA}(Z_t, Z_{t-1})$. The principal invariances of (2) are desirable for our purposes: it is invariant to isotropic rescaling of either representation, and (in the linear case) to post-multiplication by an orthogonal transform, so the signal is insensitive to benign reparameterizations of the feature space. Computationally, if $n_P \ll d$ then forming Gram matrices in $\mathbb{R}^{n_P \times n_P}$ is cheap; if $d \ll n_P$ then forming the feature covariances is cheap. Since we choose n_P small (typically 32 to 256), either route is negligible compared to any outer iteration.

Random-projection CKA for high-dimensional backbones. For large d (e.g., ViT backbones with wide embeddings) the naive computation of $\tilde{Z}^\top \tilde{Z} \in \mathbb{R}^{d \times d}$ may be unnecessarily expensive. We therefore employ a Johnson–Lindenstrauss style approximation: draw a fixed random projection $R \in \mathbb{R}^{d \times r}$ with $r \ll d$ (e.g., Gaussian or Achlioptas entries), and compute projected features $Y_t := Z_t R \in \mathbb{R}^{n_P \times r}$. We then use $s_t := \mathrm{CKA}(Y_t, Y_{t-1})$. The resulting overhead is $O(n_P d r)$ for the projection plus $O(n_P r^2)$ for the CKA computation, which we treat as o in our protocol. Since R is fixed across training, the approximation noise is stable and does not introduce spurious nonstationarity into the controller's context.

SVCCA-style alternatives and when they matter. When one seeks sensitivity to subspace changes rather than full-feature similarity, singular vector canonical correlation analysis (SVCCA) is a natural alternative. Let $\hat{U} \in \mathbb{R}^{d \times k}$ and $\hat{U}' \in \mathbb{R}^{d \times k}$ be the leading k right singular vectors of $\tilde{Z}$ and $\tilde{Z}'$, and let $\{\rho_j\}_{j=1}^k$ be the canonical correlations between the projected activations $\tilde{Z}\hat{U}$ and $\tilde{Z}'\hat{U}'$. One may define $s := \frac{1}{k}\sum_{j=1}^k \rho_j \in [0, 1]$. Exact SVCCA can be more costly than linear CKA due to SVD and CCA steps; however, with small n_P and modest k, or after random projection to r, SVCCA becomes practical. Empirically we find linear CKA sufficient for action selection, and we view SVCCA primarily as a diagnostic tool when assessing whether s_t is dominated by a low-dimensional drift.

EMA checkpoints to suppress high-frequency noise. A stability signal computed between consecutive iterates can be overly sensitive when θ_t changes rapidly due to optimizer noise or data-order effects. We therefore consider an exponential moving average (EMA) reference checkpoint $\bar{\theta}_t := \alpha\bar{\theta}_{t-1} + (1-\alpha)\theta_t$ with $\alpha \in (0, 1)$. A robust variant is

$$s_t := \mathcal{S}(Z_t, \bar{Z}_{t-1}), \qquad \bar{Z}_{t-1} := \varphi(P; \bar{\theta}_{t-1}),$$

which compares the current representation to a smoothed historical representation. This reduces variance in s_t and yields more stable gating thresholds, at the cost of maintaining one extra forward pass through the EMA model (which is still negligible for small P).

Probe batch selection and practical invariances. The probe set P should be small, fixed, and representative enough that changes in s_t correspond to global shifts rather than idiosyncratic batch effects. In practice we select P by uniform sampling from D at initialization and then freeze it. To reduce sensitivity to augmentation randomness, we either (i) store fixed augmented views for each probe example, or (ii) average features across a small, fixed number of deterministic augmentations. We emphasize that s_t is not intended to detect every meaningful training event: it is insensitive to transformations that preserve pairwise similarities in feature space (by design), and it may fail to detect changes localized to layers not probed by φ. Moreover, if representations collapse (e.g., near-constant features), CKA may appear artificially high; accordingly, we recommend monitoring simple companion statistics on P, such as feature variance $\mathrm{tr}(\tilde{Z}^\top \tilde{Z})/n_P$, to disambiguate "stable and informative" from "stable due to collapse."

Calibration diagnostics for stability informativeness. Our later near-oracle guarantee relies on s_t being informative about the relative advantage of more expensive actions. Since this is assumption-dependent, we propose a simple calibration protocol. On a sparse subset of rounds (e.g., every M steps), we execute an *audit* in which we evaluate (without updating θ) a representative cheap action $a_{\mathrm{ch}} \in \mathcal{A}_{\mathrm{cheap}}$ and a representative deep action $a_{\mathrm{dp}} \in \mathcal{A}_{\mathrm{deep}}$ on the same batch T_t, yielding an empirical gap $\hat{\Delta}_t := \ell_t(a_{\mathrm{ch}}) - \ell_t(a_{\mathrm{dp}})$. We then regress $\hat{\Delta}_t$ on s_t (e.g., isotonic regression to enforce monotonicity, or binning to obtain a reliability curve) and report the resulting prediction error as an estimate of the calibration parameter ϵ appearing in the stability-informativeness condition. This diagnostic serves two purposes: it justifies the use of stability-conditioned filtering in the controller, and it provides a principled method to set or adapt stability thresholds used by the algorithm in the next section.

5 Stability-Guided Cost-Sensitive Bandit Control

We now describe the online controller that selects, at each outer iteration t, an action $a_t \in \mathcal{A}$ encoding both (i) the task-constructor granularity (e.g., clustering resolution) and (ii) the inner-loop adaptation strategy (e.g., ANIL-style head-only steps versus MAML-style full-body steps and the number of steps). The controller observes only bandit feedback $\ell_t(a_t)$ for the executed action, but it also receives the cheap side-information scalar $s_t \in [0, 1]$ from Section 4. Our goal is to minimize the compute-regularized objective

$$\sum_{t=1}^{T} \Big(\ell_t(a_t) + \lambda c(a_t) \Big),$$

where $\lambda \geq 0$ is a user-specified trade-off parameter and $c(a) \in [1, C]$ is a known per-action cost (measured in any consistent unit such as gradient evaluations plus constructor calls).

Base controller: EXP3 on cost-regularized losses. Let $K := |\mathcal{A}|$. We define the per-round regularized loss

$$L_t(a) := \ell_t(a) + \lambda c(a), \qquad L_t(a) \in [0, 1 + \lambda C],$$

and we run an exponential-weights bandit update (EXP3-style) on L_t. The controller maintains nonnegative weights $\{w_t(a)\}_{a \in \mathcal{A}}$ and forms a sampling distribution p_t with explicit exploration. Writing $\gamma \in (0, 1]$ for the exploration rate, we set

$$p_t(a) := (1 - \gamma) \frac{w_t(a)}{\sum_{a'} w_t(a')} + \frac{\gamma}{K},$$

sample $a_t \sim p_t$, execute one outer-loop meta-update under a_t to obtain θ_{t+1}, and observe $L_t := L_t(a_t)$. We then form the usual importance-weighted estimator

$$\widehat{L}_t(a) := \frac{L_t}{p_t(a_t)} \mathbf{1}\{a = a_t\},$$

and update

$$w_{t+1}(a) := w_t(a) \exp\big(-\eta\, \widehat{L}_t(a)\big),$$

for a learning rate $\eta > 0$. If one wishes to keep losses in $[0, 1]$ for numerical stability, it suffices to rescale by $1 + \lambda C$, i.e., replace L_t by $L_t/(1 + \lambda C)$ and accordingly adjust η. This controller is intentionally agnostic to the internal structure of unsupervised meta-learning: all algorithmic complexity (constructor g, inner-loop steps for h, and outer update of f_{θ_t}) is encapsulated by the black-box action interface and its bandit feedback.

Stability-conditioned action filtering (gating). The stability signal s_t can be used to reduce wasteful exploration among clearly inappropriate compute regimes. We formalize this as a per-round eligible set $E_t \subseteq \mathcal{A}$ computed from s_t and a user-chosen rule. The distribution is then defined on E_t only:

$$p_t(a) := \begin{cases} (1 - \gamma) \dfrac{w_t(a)}{\sum_{a' \in E_t} w_t(a')} + \dfrac{\gamma}{|E_t|}, & a \in E_t, \\ 0, & a \notin E_t, \end{cases}$$

with the convention that E_t must be nonempty. A simple and practically effective instance is threshold gating. Partition $\mathcal{A} = \mathcal{A}_{\text{cheap}} \cup \mathcal{A}_{\text{deep}}$ (e.g., ANIL-like versus MAML-like). For a threshold $\tau \in [0, 1]$, one may set

$$E_t := \begin{cases} \mathcal{A}_{\text{cheap}}, & s_t \geq \tau, \\ \mathcal{A}_{\text{deep}}, & s_t < \tau, \end{cases}$$

which encodes the design heuristic that high representation stability indicates that expensive inner adaptation is less likely to be beneficial. More graded policies are also natural: with multiple thresholds $1 \geq \tau_1 > \tau_2 > \cdots$, one may define a ladder of eligible sets of increasing cost, thereby restricting the controller to progressively deeper actions only when instability persists.

We emphasize two implementation details. First, to avoid pathological exclusion due to noisy s_t, we may enforce a safety fallback $E_t = \mathcal{A}$ whenever the chosen rule yields $|E_t| < m_{\min}$ for a small $m_{\min} \geq 2$, ensuring meaningful exploration. Second, gating need not be hard; one can implement *soft gating* by multiplying weights by a stability-dependent prior $\pi_t(a) \in [0, 1]$ (e.g., $\pi_t(a)$ decreasing with $c(a)$ when s_t is high), and sampling from the renormalized product $\pi_t(a)w_t(a)$. This retains a nonzero probability for all actions while still biasing the search toward plausible compute regimes.

Two-stage (hierarchical) selection. When $\mathcal{A}$ is large and structured, a two-stage controller is convenient. We introduce a coarse regime variable $r_t \in \{\text{cheap}, \text{deep}\}$ and a within-regime action $a_t \in \mathcal{A}_{r_t}$. The first stage can be implemented either deterministically from s_t (thresholding) or via a small bandit on the two regimes using the same loss L_t but with the regime costs defined as $\min_{a \in \mathcal{A}_r} c(a)$. Conditional on r_t, we then run an independent EXP3 instance over $\mathcal{A}_{r_t}$. This decomposition reduces variance in the within-regime estimates and makes it straightforward to add or remove fine-grained actions without retuning a single flat K-arm controller.

Contextual variant via discretized stability. Although s_t is a scalar, it can be used as context by maintaining separate bandit weights for different stability ranges. Concretely, fix bins $\{I_b\}_{b=1}^{B}$ partitioning $[0, 1]$ and let $b(t)$ be the bin index containing s_t. We maintain weights $w_t^{(b)}(a)$ for each bin and update only the active bin:

$$w_{t+1}^{(b(t))}(a) = w_t^{(b(t))}(a) \exp\big(-\eta \widehat{L}_t(a)\big), \qquad w_{t+1}^{(b)}(a) = w_t^{(b)}(a) \text{ for } b \neq b(t).$$

Sampling is performed from p_t computed using $w_t^{(b(t))}$ (and optionally gating). This "binned contextual bandit" is a minimal mechanism for conditioning on stability without assuming any parametric relationship between s_t and loss gaps.

Budget handling and adaptive λ. While our primary presentation uses the Lagrangian objective, practitioners often specify a target average cost $\bar{c}$. A standard remedy is to update λ online by dual ascent:

$$\lambda_{t+1} := \Big[\lambda_t + \rho\big(c(a_t) - \bar{c}\big)\Big]_+,$$

with step size $\rho > 0$ and projection onto $[0, \infty)$. The controller then uses λ_t in place of a fixed λ when forming L_t. This converts the trade-off into an adaptive constraint-tracking mechanism without altering the bandit feedback interface.

Summary pseudocode. For clarity we collect the preceding choices into the following template:

> Initialize $w_1(a) = 1$ for all a. For $t = 1, \ldots, T$: compute s_t; choose E_t from s_t (optional); form p_t from w_t restricted to E_t with exploration γ; sample $a_t \sim p_t$; execute one meta-update under a_t; observe $\ell_t(a_t)$ and incur cost $c(a_t)$; set $L_t = \ell_t(a_t) + \lambda c(a_t)$; update weights by importance weighting; optionally update λ by dual ascent.

In the next section we analyze the regret of these controllers under bandit feedback and quantify when stability-conditioned filtering yields near-oracle compute selection.

6 Regret Guarantees and Near-Oracle Compute Selection

We analyze the stability-guided controller of Section 5 as an online learning algorithm over the finite action set $\mathcal{A}$ under bandit feedback. Recall that the meta-learner incurs at outer iteration t the query loss $\ell_t(a_t) \in [0, 1]$ for the executed action a_t, and we regularize by the known cost $c(a_t) \in [1, C]$ through the Lagrangian loss

$$L_t(a) \;:=\; \ell_t(a) + \lambda c(a) \in [0, 1 + \lambda C].$$

The controller uses only the realized scalar $L_t = L_t(a_t)$ and the sampling probability $p_t(a_t)$ to form an importance-weighted estimate. For the present section we treat the inner/outer optimization and task construction as black boxes that merely instantiate the per-round loss vector $L_t(\cdot)$.

Theorem 1 (adversarial regret for cost-regularized EXP3). Assume that for each t, the loss vector $L_t(\cdot)$ is arbitrary with $L_t(a) \in [0, 1 + \lambda C]$. Run the base controller (no gating) with exploration $\gamma \in (0, 1]$ and learning rate $\eta > 0$ on the scaled losses $\widetilde{L}_t(a) := L_t(a)/(1 + \lambda C) \in [0, 1]$. Then the expected regret with respect to the best fixed action in hindsight satisfies

$$\mathbb{E}\left[\sum_{t=1}^{T} L_t(a_t)\right] - \min_{a \in \mathcal{A}} \sum_{t=1}^{T} L_t(a) \;\le\; O\Big((1 + \lambda C)\sqrt{TK \log K}\Big), \qquad K := |\mathcal{A}|.$$

In particular, up to logarithmic factors, the controller pays a $\sqrt{TK}$ price for online selection among K possible inner-loop/constructor configurations, while the cost regularization contributes only via the scale factor $1 + \lambda C$.

Proof sketch. We apply the standard EXP3 potential argument to $\widetilde{L}_t$. The importance-weighted estimator $\widehat{\widetilde{L}}_t(a) = (\widetilde{L}_t / p_t(a_t))\mathbf{1}\{a = a_t\}$ is unbiased for $\widetilde{L}_t(a)$. Bounding the one-step increase of the log-partition $\log \sum_a w_t(a)$ yields a comparison between the algorithm's cumulative estimated loss and that of any fixed comparator action, plus a variance term controlled by the explicit exploration floor $p_t(a) \geq \gamma/K$. Choosing η and γ on the order of $\sqrt{(\log K)/(TK)}$ gives the stated bound after rescaling by $1 + \lambda C$.

Theorem 2 (piecewise-stationary comparator). In many meta-training regimes the optimal compute/action choice drifts over time (e.g., as pseudo-label quality improves). Suppose the identity of the best action changes at most S times across T rounds, yielding an unknown partition of $\{1, \ldots, T\}$ into $S + 1$ contiguous segments. Using a restarted variant of the controller (or an EXP3.S-style algorithm), we obtain the dynamic-regret guarantee

$$\mathbb{E}\left[\sum_{t=1}^{T} L_t(a_t)\right] - \min_{\substack{\text{segment actions} \\ a^{(1)},\ldots,a^{(S+1)}}} \sum_{j=1}^{S+1} \sum_{t \in I_j} L_t(a^{(j)}) \;\leq\; \widetilde{O}\Big((1+\lambda C)\sqrt{TK(S+1)}\Big),$$

where I_j are the (unknown) segments and $\widetilde{O}$ hides polylogarithmic factors. Thus, if the best regime changes infrequently, the controller tracks it with only a $\sqrt{S+1}$ multiplicative overhead relative to the stationary case.

Theorem 3 (stability-informativeness $\Rightarrow$ near-oracle compute selection). We now formalize when stability-based gating can be more than a heuristic. Partition $\mathcal{A} = \mathcal{A}_{\text{cheap}} \cup \mathcal{A}_{\text{deep}}$, where the former correspond to low-cost inner adaptations and the latter to higher-cost adaptations. Assume there exists a measurable function $\Delta : [0, 1] \to \mathbb{R}$ and calibration error $\epsilon \geq 0$ such that, for every t,

$$\Big| \min_{a \in \mathcal{A}_{\text{cheap}}} \ell_t(a) \;-\; \min_{a \in \mathcal{A}_{\text{deep}}} \ell_t(a) \;-\; \Delta(s_t) \Big| \;\leq\; \epsilon.$$

Define the cost gap $\Delta c := \min_{a \in \mathcal{A}_{\text{deep}}} c(a) - \min_{a \in \mathcal{A}_{\text{cheap}}} c(a) > 0$. Consider a gating rule that admits only cheap actions when $\Delta(s_t) \leq \lambda \Delta c$ (deep compute not justified in the Lagrangian sense) and admits deep actions otherwise. Then the expected Lagrangian suboptimality relative to an oracle that observes $\ell_t(a)$ for all a is bounded by

$$O(\epsilon T) \;+\; (\text{bandit regret incurred within the eligible sets}).$$

In words, if the stability signal predicts the cheap-versus-deep loss gap up to error ϵ, then stability-gated selection loses at most $O(\epsilon)$ per round compared

to a compute-aware oracle, plus the unavoidable bandit cost of choosing among actions within the selected regime.

Proof sketch. The oracle's per-round decision compares $\min_{a \in \mathcal{A}_{\text{cheap}}} \ell_t(a) + \lambda \min_{a \in \mathcal{A}_{\text{cheap}}} c(a)$ to the analogous deep quantity. The gating rule makes the same comparison but with $\Delta(s_t)$ substituted for the unknown loss gap; the assumed calibration inequality implies that whenever the rule selects the wrong regime, the resulting excess Lagrangian loss is at most $O(\epsilon)$. Summing over t yields the $O(\epsilon T)$ term, and the remaining difference is controlled by applying Theorem 1 (or Theorem 2) to the restricted action set active under gating.

Theorem 4 (matching lower bound). The $\sqrt{TK}$ dependence in Theorem 1 is information-theoretically unavoidable in the worst case. Specifically, for any online controller with bandit feedback over K actions and losses in $[0, 1]$, there exists an adversarial loss sequence such that the expected regret against the best fixed action is $\Omega(\sqrt{TK})$. This remains true even if the controller additionally observes a side-information signal s_t that is independent of the losses (or otherwise non-informative). Consequently, improvements over $\Theta(\sqrt{TK})$ require additional structure beyond adversarial bandits, such as stochastic losses, realizable contextual models, or a stability signal satisfying an informativeness condition of the form used in Theorem 3.

The foregoing theorems provide our main guarantees: standard adversarial regret for cost-regularized action selection, extensions to nonstationarity, and a precise route by which representation stability can justify aggressive compute reduction without sacrificing accuracy. We next quantify the computational overhead of stability measurement and controller maintenance, and we discuss optimality and sensitivity to noisy stability in the complexity and information-theoretic terms.

Complexity, overhead, and scaling. We separate the computational cost of the controlled meta-update from the overhead of the controller and stability measurement. At round t, executing the chosen action a_t incurs the dominant cost $c(a_t)$, which aggregates (i) task construction cost (e.g., clustering, Sinkhorn, or assignment updates at the granularity encoded by a_t), (ii) the prescribed number of inner-loop gradient steps and associated forward/backward passes, and (iii) the outer/meta-gradient computation. The controller adds only: (a) an EXP3-style weight update and sampling step over $K = |\mathcal{A}|$ arms, and (b) the stability computation $s_t = S(\theta_t, \theta_{t-1}; P)$ on a fixed probe batch P. The controller arithmetic is $O(K)$ time and $O(K)$ memory per iteration, which is negligible compared to a single gradient step once the backbone has nontrivial size.

Stability computation cost and dependence on $|P|$. Let $|P|$ denote the probe batch size, d the representation dimension at the probed layer, and $r \ll d$ the rank of a random projection used to approximate CKA/SVCCA-type similarities. A typical implementation computes probe representations $Z_t \in \mathbb{R}^{|P|\times d}$ and Z_{t-1} by two forward passes (no backpropagation), projects to $Z'_t = Z_t R \in \mathbb{R}^{|P|\times r}$ for a fixed $R \in \mathbb{R}^{d\times r}$, and then evaluates a normalized similarity. This yields overhead

$$o \;=\; O\big(|P| \cdot \mathrm{FWD}(f_\theta)\big) \;+\; O\big(|P|dr\big) \;+\; O\big(|P|r^2\big),$$

where $\mathrm{FWD}(f_\theta)$ denotes the cost of a single forward pass through the backbone. In regimes where inner-loop actions require multiple forward/backward passes, it is typical that $o \ll c(a)$ for all $a \in \mathcal{A}$; indeed, if $c(a)$ is measured in gradient evaluations, then o is comparable to at most a small constant number of forward evaluations. The probe size $|P|$ controls a variance–cost trade-off: increasing $|P|$ decreases the noise of s_t but raises overhead linearly. In practice, $|P|$ can be fixed to a small fraction of the training batch size, and we may also amortize stability by computing it only every m rounds, replacing s_t by an exponential moving average $\bar{s}_t$ with negligible effect on the controller analysis provided the gating rule uses the same signal used in practice.

Total time and space over T rounds. Summing per-round costs, the overall runtime is

$$\sum_{t=1}^{T} \big(c(a_t) + o\big) \;=\; \sum_{t=1}^{T} c(a_t) \;+\; To.$$

The first term is the intended controlled compute, while the second term is an additive overhead independent of the selected action. Controller state requires storing weights $w_t(a)$ (and optionally their normalized probabilities $p_t(a)$), thus $O(K)$ memory. Stability computation can be performed either by caching the projected probe features $Z'_t \in \mathbb{R}^{|P|\times r}$, giving $O(|P|r)$ extra memory, or by recomputing both Z'_t and Z'_{t-1} on demand, which uses $O(|P|r)$ transient memory but incurs two forward passes. In both cases the memory is negligible relative to standard optimizer states for the backbone (e.g., Adam moments), which scale with the number of parameters.

Dependence on the number of actions K. The dependence on K enters in two places: (i) the controller update is linear in K, and (ii) the regret bounds scale as $\tilde{O}(\sqrt{TK})$ under adversarial losses (Theorem 1). Consequently, K should reflect the granularity at which we actually need distinct inner-loop strategies. In our setting, actions typically factor into a small Cartesian product (e.g., #inner steps $\times$ constructor resolution $\times$ whether

higher-order gradients are used), and thus K is moderate. If one wishes to expose a large discrete grid of actions, then it is natural to exploit structure (e.g., hierarchical or factorized bandits) to replace the $\sqrt{K}$ dependence by sums of smaller terms; this is an algorithmic refinement rather than a change in the underlying bilevel learning problem.

Optimality and what cannot be improved in the worst case. Even though the controller observes a side-information signal s_t, the bandit feedback model still restricts us to observing $\ell_t(a_t)$ only for the chosen action. Theorem 4 formalizes that, without further assumptions linking s_t to the loss vectors $L_t(\cdot)$, the minimax regret scales as $\Omega(\sqrt{TK})$. Thus, improvements in the dependence on T and K are information-theoretically impossible under adversarial losses unless we either (a) assume stochasticity/realizability, (b) enrich the feedback (e.g., observe losses for multiple actions per round), or (c) assume that the stability signal is genuinely informative about loss differences, as in Theorem 3. In particular, our cost regularization through $\lambda c(a)$ changes only the loss scale $1+\lambda C$, and does not alter the fundamental exploration cost: identifying a near-optimal action among K candidates still requires $\Theta(\sqrt{TK})$-type uncertainty under bandit feedback.

Sensitivity to stability noise and miscalibration. The stability signal is computed from a finite probe and is therefore noisy. We may model this by observing $\widetilde{s}_t = s_t + \xi_t$ where ξ_t is bounded or subgaussian, and by allowing the calibration condition in Theorem 3 to hold with an effective error ϵ_{eff} that absorbs both approximation error of S (e.g., random-projection CKA) and measurement noise (finite $|P|$). Under such a model, the near-oracle term $O(\epsilon T)$ becomes $O(\epsilon_{\text{eff}} T)$, and the dominant failure mode is systematic misgating: repeatedly excluding the regime (cheap or deep) that would minimize the Lagrangian. Two mitigations are immediate and preserve the bandit analysis within the eligible set: (i) *soft gating*, in which we do not set $p_t(a) = 0$ but merely downweight disfavored regimes, maintaining a nonzero exploration floor across all actions; and (ii) *hysteresis/EMA gating*, in which the gate depends on $\bar{s}_t$ and uses separate thresholds for switching from cheap to deep and vice versa, reducing spurious oscillations when s_t is near the decision boundary $\Delta(s) = \lambda \Delta c$. Both mechanisms trade a small amount of additional compute for robustness, and can be viewed as controlling the number of rounds on which the gate disagrees with the oracle regime, which is precisely the quantity that enters the approximation term discussed after Theorem 3.

Compute-budget variants and tuning. While our analysis is phrased in the Lagrangian form $\sum_t \ell_t(a_t) + \lambda c(a_t)$, the same complexity accounting applies to the constrained budget $\sum_t c(a_t) \le B$. In practice one may adapt

λ online via a dual update to target a desired average cost, which adds only constant additional arithmetic per round. From the standpoint of overhead, this does not change the leading terms: stability measurement remains To and controller maintenance remains $O(TK)$, while the meta-update cost remains $\sum_t c(a_t)$, which is the quantity being optimized.

Summary. The controller introduces negligible memory and $O(K)$ time per round, and stability measurement contributes an additive To overhead linear in probe size and (projected) representation dimension. The $\sqrt{TK}$ dependence of regret is minimax-optimal in the absence of further structure, so any practical gains must come from either (i) the cost scaling captured by λ, which encourages cheaper actions when accuracy permits, or (ii) informativeness of stability, which can reduce unnecessary exploration of expensive regimes while remaining consistent with the online bandit setting.

Implementation and experimental plan. We implement SG-MetaControl as a thin wrapper around an existing unsupervised bilevel meta-learning codebase with in-the-loop task construction (DHM-UHT-style). Concretely, each outer round t samples a batch $T_t \subset D$, applies the action-selected constructor $g(\cdot; a_t)$ to obtain pseudo-partitions (e.g., cluster assignments, Sinkhorn transport plans, or nearest-neighbor graphs), splits the batch into support/query subsets according to the DHM-UHT protocol, reinitializes the task module h, performs the inner adaptation specified by a_t (e.g., ANIL-like head-only steps versus MAML-like full-body steps, with optional higher-order gradients), and then applies the outer update to θ. The per-round bandit loss $\ell_t(a_t)$ is taken to be the query/meta-loss produced by that round's construction and adaptation, clipped to $[0, 1]$ by a fixed affine rescaling determined from a short calibration run (the same rescaling is applied across all actions to maintain a common range). Compute costs $c(a)$ are measured in a hardware-agnostic proxy (number of forward/backward passes plus constructor calls) and are optionally validated against wall-clock time; we report both when available.

Action set design and cost calibration. We define $\mathcal{A}$ as a small Cartesian product capturing the dominant compute–accuracy knobs in DHM-UHT. A representative choice is

$$\mathcal{A} = \{0, 1, 2, 4\} \text{ inner steps} \times \{\text{head-only}, \text{full-body}\} \times \{\text{coarse}, \text{fine}\} \text{ constructor granularity},$$

yielding $K = 16$ actions. The "coarse/fine" granularity may correspond to a smaller/larger number of clusters, Sinkhorn iterations, or graph neighborhood size. For each action a, we precompute $c(a)$ by counting primitive operations (one clustering call, one forward pass, one backward pass), and we additionally record empirical wall-clock ratios on a fixed reference batch;

in the controller we use the proxy $c(a)$ to preserve portability across devices, while experimental plots may annotate both. We fix λ either by targeting a desired average cost $\frac{1}{T}\sum_t c(a_t)$ via a dual update, or by sweeping λ over a logarithmic grid to obtain compute–accuracy frontiers.

Stability signal and gating instantiation. We instantiate $s_t = S(\theta_t, \theta_{t-1}; P)$ using random-projection linear CKA on a single probed layer (typically the penultimate block), computed on a fixed probe batch P resampled once at initialization and then held fixed. We use an exponential moving average $\bar{s}_t$ for gating and log both s_t and $\bar{s}_t$ to quantify noise. The optional eligible-set rule is chosen to match Theorem 3's cost-benefit comparison: we partition $\mathcal{A}$ into $\mathcal{A}_{\text{cheap}}$ (head-only and/or fewer inner steps and/or coarse constructors) and $\mathcal{A}_{\text{deep}}$ (full-body and/or more steps and/or fine constructors), and we switch regimes based on whether $\bar{s}_t$ lies above or below a threshold. To reduce oscillations we employ hysteresis (two thresholds $\tau_\uparrow < \tau_\downarrow$) and soft gating (never setting probabilities to zero, but enforcing a small exploration floor α/K on all arms). This permits a direct empirical study of the trade-off between stability-driven compute savings and the risk of misgating.

Baselines and ablations. We evaluate against (i) fixed-action training for each $a \in \mathcal{A}$, yielding an oracle-in-hindsight reference for the Lagrangian objective and a compute–accuracy Pareto set; (ii) uniform random action selection; (iii) loss-only bandit control (EXP3 on ℓ_t without stability gating); (iv) stability-only heuristics (deterministic gating between a single cheap and a single deep action based on $\bar{s}_t$, without EXP3 within the regime); and (v) cost-blind bandit control (EXP3 on ℓ_t with $\lambda = 0$), which isolates the effect of explicit compute regularization. We also ablate the stability estimator (CKA versus cosine similarity of mean features; different probe sizes $|P|$ and projection ranks r) to test robustness of the controller to measurement noise, as suggested by the discussion following Theorem 3. Finally, we include a "no-constructor-change" ablation in which g is fixed (no clustering updates) to verify that the controller's benefits indeed align with constructor drift rather than generic training nonstationarity.

Reproducing DHM-UHT settings. Our first experimental block reproduces DHM-UHT training schedules, data augmentations, and evaluation pipelines on the standard benchmark suite used in that literature (same datasets, episodic sampling rules, and evaluation splits). We report (a) downstream few-shot accuracy (e.g., N-way K-shot episodes using a standard ridge-regression or prototype head on frozen features), (b) linear-probe accuracy on a held-out labeled set, and (c) compute statistics: average $c(a_t)$, fraction of deep actions, and achieved Lagrangian objective $\sum_t(\ell_t(a_t) + \lambda c(a_t))$. To connect to the online-learning framing, we also plot the cumulative regret

proxy relative to the best fixed action in hindsight (computed offline from separate runs per action), acknowledging that this is an empirical quantity rather than an observed online signal.

Scaling to larger backbones and longer horizons. We then scale the same controller to larger representation learners (e.g., ResNet-50 and ViT-S/ViT-B) where the separation between cheap and deep inner-loop updates is magnified in wall-clock time. The action set is adapted to include mixed-precision and gradient-checkpointing options as additional compute levers when appropriate; these are treated as actions with associated costs and (potentially) different loss profiles. We test longer horizons T to examine nonstationarity effects and, when relevant, we include the restarted/nonstationary controller variant corresponding to Theorem 2. The central question is whether stability-guided control continues to select predominantly cheap actions once representations stabilize, while still escalating compute during representation shifts (e.g., at learning-rate drops or after constructor re-initializations).

Controlled nonstationarity: label noise, heterogeneity, and cluster drift. To isolate mechanisms, we construct three controlled regimes. (i) *Pseudo-label noise*: we inject controlled corruption into the constructor output (e.g., flipping a fraction ρ of assignments, perturbing transport plans, or adding Gaussian noise before clustering) and measure how the controller shifts toward deep updates as ρ increases. (ii) *Heterogeneity*: we form D as a mixture of domains (or classes with differing augmentation difficulty) and schedule domain proportions over time; we test whether stability drops at mixture shifts and whether compute allocation responds accordingly. (iii) *Cluster drift*: we explicitly change constructor granularity over time (e.g., increasing the number of clusters or annealing Sinkhorn temperature), inducing predictable changes in the pseudo-task distribution; this setting is designed to validate that s_t detects representation/assignment mismatches and that deep actions are selected primarily when the induced drift makes them cost-effective.

Reporting and reproducibility. We run all methods with matched outer-loop optimization hyperparameters and report means and standard errors over multiple seeds. We log per-round ℓ_t, $c(a_t)$, s_t, and p_t to enable post hoc verification of controller behavior (e.g., whether action probabilities concentrate as stability increases). We additionally provide plots of compute–accuracy frontiers as λ varies, and we report sensitivity to $|P|$, r, γ, and η, emphasizing whether qualitative conclusions persist under reasonable retuning. These experiments jointly test (a) the empirical relevance of stability-informativeness, (b) the practical impact of cost regularization, and (c)

the extent to which the controller attains near-oracle compute allocation in regimes where stability reliably anticipates the cheap-versus-deep loss gap.

Limitations and extensions. Our formulation deliberately restricts the controller to a finite action set $\mathcal{A}$, which is appropriate when the dominant design choices are categorical (e.g., head-only versus full-body, coarse versus fine constructors) or when a small grid over inner-loop steps suffices. A first limitation is that several practically important knobs are naturally continuous or high-resolution, such as clustering hyperparameters (e.g., ε in DBSCAN, min_samples, Sinkhorn entropic temperature, augmentation strength), inner-loop learning rates, or the fraction of layers adapted. Discretization of these quantities into $\mathcal{A}$ can induce either a large K (hurting the $\sqrt{K}$ dependence) or a coarse grid that misses the compute–accuracy sweet spot.

A direct extension is to replace finite-armed bandits by *continuous* (or large) action models with structure. One mathematically clean route is to endow the action space with a metric and assume a regularity condition on the cost-regularized loss $L_t(a) = \ell_t(a) + \lambda c(a)$, such as Lipschitzness in a for each t, or stochastic smoothness in expectation. Then we may apply continuum-armed bandit techniques (e.g., zooming-style adaptive discretizations) to obtain regret scaling with an intrinsic dimension rather than with K. In our setting, one would represent actions as vectors

$$a = (k,\ \texttt{scope},\ r,\ \alpha,\ \xi, \ldots)$$

encoding inner steps k, adaptation scope, constructor resolution r, inner-loop step size α, and constructor hyperparameters ξ. The principal difficulty is that the map $a \mapsto \ell_t(a)$ need not be smooth under adversarial task sequences, and thus any improved rate demands explicit structural assumptions (e.g., stochastic tasks and stable constructor behavior). A more conservative alternative, which preserves adversarial guarantees, is hierarchical or coarse-to-fine discretization: we begin with a small $\mathcal{A}_0$, periodically refine around empirically good actions, and treat refinement as increasing K over time with a corresponding nonstationary analysis.

A second limitation is that our contextual signal is essentially one-dimensional (stability s_t, possibly smoothed), and the controller is a simple exponential-weights scheme (with optional gating). This choice is intentional: it keeps the update unbiased and the analysis standard. However, practical training dynamics offer richer side information, such as $\ell_{t-1}(a_{t-1})$, gradient norms, variance proxies from augmentation views, entropy of pseudo-label assignments, or disagreement between multiple constructors. We may therefore consider *learned contextual policies* that map a feature vector $x_t \in \mathbb{R}^d$ (including s_t) to a distribution over actions. If we posit a realizable model class Π (e.g., small MLPs producing logits over actions), then the controller becomes a contextual bandit problem with bandit feedback. Algorithmically,

one may use EXP4-style aggregation over a finite policy class, or (for parametric Π) online methods such as neural contextual bandits with importance-weighted losses. Analytically, regret guarantees depend on the complexity of Π (e.g., $\log|\Pi|$ in EXP4) or on realizability assumptions (e.g., linear payoffs in LinUCB). In exchange, the controller can learn context-dependent schedules, such as "use deep actions during constructor entropy spikes" without manually setting thresholds.

This direction raises a methodological caveat: as soon as we learn a contextual mapping $x_t \mapsto p_t$, we must address exploration in a state-dependent manner to preserve identifiability, and we must control bias introduced by function approximation. In particular, if a learned policy collapses too early, then some actions may never be sampled in contexts where they matter, invalidating both regret claims and empirical conclusions. A principled mitigation is to enforce a uniform exploration floor $p_t(a) \geq \alpha/K$ (as we already do for soft gating) and to use doubly robust estimators for off-policy evaluation when comparing candidate controllers.

A third limitation concerns *multi-objective* compute constraints. Our objective $\sum_t \ell_t(a_t) + \lambda c(a_t)$ captures a single scalarized notion of cost. In practice, one may face multiple budgets simultaneously: wall-clock latency, peak memory, energy, communication in distributed training, or even a constraint on stability disruption itself (e.g., limiting representation drift to preserve downstream compatibility). A natural extension is to consider vector-valued costs $c^{(j)}(a)$ for $j = 1, \ldots, m$ and either (i) a scalarization with multiple multipliers λ_j, giving $L_t(a) = \ell_t(a) + \sum_j \lambda_j c^{(j)}(a)$, or (ii) a constrained formulation

$$\min \sum_{t=1}^{T} \ell_t(a_t) \quad \text{s.t.} \quad \sum_{t=1}^{T} c^{(j)}(a_t) \leq B_j, \ \forall j.$$

The latter suggests primal–dual updates in which λ_j are adapted online via subgradient ascent on constraint violations, coupled with an EXP3-like inner loop on the instantaneous Lagrangian. The technical point is that adversarial bandit analysis extends cleanly when λ_j are predictable and bounded, but care is required when λ_j are updated from bandit feedback, since the dual variables become data-dependent and can amplify estimator variance.

Relatedly, one may wish to optimize a *risk-sensitive* objective rather than the mean query loss, e.g., controlling tail events where pseudo-labeling collapses. This leads to bandit objectives involving CVaR or robust losses, which again can be addressed by online convex optimization with bandit feedback, at the cost of additional assumptions (boundedness, mixing) and typically worse constants. From the meta-learning perspective, these formulations correspond to explicitly pricing instability events rather than using stability only as a heuristic input to the controller.

A fourth limitation is *transfer* of controllers across datasets, backbones, and horizons. SG-MetaControl is, by design, an online algorithm that can

start from uniform weights and adapt within a single run. Nevertheless, in large-scale practice we often repeat training across many datasets or model sizes, and it is natural to ask whether a controller tuned on one setting can warm-start another. In the finite-action case, a simple mechanism is prior initialization $w_1(a) \propto \exp(-\beta \hat{L}(a))$ based on historical performance $\hat{L}(a)$ from related runs, yielding faster concentration when the transfer is valid. More structured transfer treats the action losses as arising from a latent environment parameter ϕ (dataset/backbone descriptor) and learns a mapping $\phi \mapsto$ prior over actions, which is then refined online. This is essentially a meta-contextual bandit problem: we learn a controller *of controllers*. The main open issue is robustness: if transfer is poor, an overly confident prior can delay exploration and incur large transient regret. Thus any transferable controller should retain explicit mechanisms for uncertainty (e.g., entropy regularization or posterior sampling) and must guarantee recovery to baseline exploration in the worst case.

Finally, our use of stability s_t relies on a fixed probe distribution and on a particular layer choice, and it is not guaranteed to be informative in all regimes. For example, stability may saturate even when pseudo-tasks shift in a way that matters for adaptation, or it may fluctuate due to benign symmetries (e.g., feature rotations) that do not affect downstream performance. Extensions here include: (i) multi-layer stability vectors $s_t^{(\ell)}$ to disambiguate local versus global changes; (ii) task-constructor stability metrics (e.g., cluster-assignment mutual information across rounds) that more directly measure pseudo-label drift; and (iii) learned stability predictors trained to forecast the cheap-versus-deep loss gap. Each of these moves the signal closer to what Theorem 3 assumes, but also introduces additional modeling choices whose failure modes must be empirically and theoretically characterized.

Conclusion: stability-guided self-tuning meta-learning. We have studied an unsupervised bilevel meta-learning loop in which the learner must repeatedly construct pseudo-tasks from unlabeled data and decide, online, how much inner-loop adaptation and task-constructor granularity to spend on each outer iteration. The central observation is that, in such systems, *compute is itself a control variable*: the difference between an ANIL-like update and a MAML-like update, or between coarse and fine pseudo-label partitions, is not merely an engineering detail but a decision that trades immediate progress against cost. Our contribution is to cast this decision as an online learning problem with bandit feedback and to show that a simple controller, driven by exponential weights on a cost-regularized loss, yields provable performance guarantees while remaining compatible with the stochastic and nonconvex nature of the underlying meta-optimizer.

Concretely, we define the per-round cost-regularized objective

$$L_t(a) \;=\; \ell_t(a) + \lambda c(a),$$

where $\ell_t(a) \in [0,1]$ is the query/meta-loss incurred by choosing action $a \in \mathcal{A}$ at outer iteration t, $c(a) \in [1, C]$ is a known compute proxy, and $\lambda \geq 0$ controls the compute–accuracy trade-off. The controller selects a_t online under bandit feedback and updates via an EXP3-style importance-weighted estimator. This yields an adversarial regret bound scaling as $\tilde{O}((1+\lambda C)\sqrt{TK})$ against the best fixed action in hindsight (Theorem 1), extensions to nonstationary comparators (Theorem 2), and a matching minimax lower bound $\Omega(\sqrt{TK})$ (Theorem 4). The latter is not merely a technicality: it formalizes that, without additional structure, no controller can reliably outperform the $\sqrt{T}$ exploration barrier, even if it observes arbitrary side information uncorrelated with losses. Thus, any practical improvement beyond minimax rates must exploit *informativeness* of observed signals.

The role of representation stability is precisely to supply such exploitable structure while retaining negligible overhead. We compute a stability statistic $s_t = S(\theta_t, \theta_{t-1}; P) \in [0,1]$ from a fixed probe batch P, chosen to be small enough that its cost o is dominated by every action cost $c(a)$. Empirically, in unsupervised task construction, instability often coincides with pseudo-label drift, abrupt changes in partition structure, or feature reshaping induced by the outer update; these are precisely the regimes in which deeper adaptation or higher-resolution constructors tend to pay off. This intuition is made explicit by the stability-informativeness condition in Theorem 3, which relates the stability level s_t to the gap between the best achievable losses among cheap versus deep actions. Under this condition, a stability-gated strategy obtains near-oracle compute selection up to an additive calibration term $O(\epsilon T)$, separating the cost of *learning* within an action class (handled by bandit regret) from the cost of *deciding* which class is worth the compute (handled by the informativeness error ϵ).

From a systems perspective, the resulting picture is a self-tuning meta-learning procedure with two nested feedback loops: the inner loop adapts a task-specific module h according to the selected strategy; the outer loop updates the shared representation f_θ; and the controller modulates the inner-loop depth and constructor granularity so as to minimize a compute-aware objective over the training horizon. Importantly, the controller is agnostic to the details of the underlying bilevel optimizer: it treats each candidate configuration as an arm and only requires that each arm can be executed for one outer iteration and returns a bounded query loss. This modularity is essential in practice, because the space of plausible unsupervised meta-learning recipes continues to expand (new constructors, new augmentations, new normalization and regularization techniques), while the controller interface remains stable: a set of actions with known costs, a bandit loss, and a cheap stability signal.

We emphasize a practical implication that becomes decisive at 2026 scale. As backbones grow and training pipelines move toward longer horizons and more heterogeneous data streams, the marginal cost of deep inner-loop adap-

tation rises substantially, not only in raw FLOPs but also in memory pressure, communication overhead in distributed settings, and wall-clock variance. In such regimes, static choices (always deep, always shallow, always fine-grained) are structurally mismatched to the temporal heterogeneity of unlabeled training. One should instead expect *phases*: stable phases where representation change is modest and cheap actions suffice, and volatile phases where pseudo-task definitions shift and deeper adaptation prevents collapse or accelerates recovery. Stability-guided control provides a principled mechanism for this phase adaptation, with the guarantee that, even under adversarial loss sequences, the controller cannot be forced into arbitrarily poor long-run performance relative to the best constant compute strategy, up to the unavoidable $\sqrt{TK}$ term.

We also draw a methodological conclusion. In unsupervised meta-learning, the absence of labels makes it tempting to introduce increasingly elaborate heuristics for task construction and inner-loop scheduling. Our analysis suggests a different discipline: whenever we introduce a heuristic signal (here, stability), we should ask whether it can be (i) computed cheaply and repeatedly, and (ii) related, even approximately, to a decision-relevant quantity (here, the cheap-versus-deep loss gap). When such a relation holds, Theorem 3 shows that the signal can be elevated from a diagnostic to a control primitive with a quantifiable suboptimality cost. When it does not hold, Theorem 4 reminds us that no amount of cleverness can circumvent the fundamental bandit uncertainty without further assumptions.

In summary, the stability-guided controller gives a mathematically transparent and implementation-friendly bridge between two desiderata that are usually in tension: robust online selection under bandit feedback, and aggressive compute savings via conditional escalation. The resulting algorithm is not an alternative to unsupervised meta-learning; it is a *wrapper* that makes unsupervised meta-learning self-tuning with respect to compute. We regard this as an enabling ingredient for the next generation of unsupervised adaptation systems: ones that must operate over long horizons, under shifting pseudo-task definitions, with limited budgets, and at scales where a fixed "deep everywhere" rule is no longer tenable. The open technical agenda is correspondingly clear: to sharpen the stability–performance link in more realistic stochastic environments, to characterize when stability is predictive versus merely correlated, and to design controllers whose guarantees remain meaningful while the underlying meta-learner and task constructor evolve.