Subgoal-Augmented HRM: Directional Latent Subgoals for Temporal Abstraction

Abstract

We investigate whether reasoning can be improved by explicitly organizing computation in latent space. We propose Subgoal-Augmented HRM, a feudal-style extension of the Hierarchical Reasoning Model (HRM) in which a slow high-level module periodically emits a normalized directional subgoal encoding medium-horizon intent. The subgoal guides computation by biasing hierarchical hidden-state updates (optionally via a learned commitment gate) and by inducing a cosine-based alignment loss that encourages goal-directed latent trajectories. On ConceptARC/ARC-mini, we find a clear optimum at $\lambda = 0.05$ (with manager period $P = 4$), and observe that moderate update periods ($P \in \{3, 6\}$) outperform both overly frequent updates ($P = 1$) and longer horizons, indicating a stability– adaptivity tradeoff. Across five replications at $\lambda = 0.05$, $P = 3$ slightly improves LM loss over $P = 4$ (1.5946 vs. 1.6013), suggesting directional latent subgoals act as a lightweight planning prior that geometrically shapes multi-step computation.

1 Introduction

Modern neural reasoning systems often rely on fixed-depth forward computation augmented by explicit token-level decomposition strategies such as Chain-of-Thought prompting (Wei et al., 2022). While effective in some domains, these approaches can be brittle to step ordering, produce long intermediate traces, and incur high inference latency. Recent work has therefore explored latent reasoning architectures, in which multi-step computation occurs within hidden-state space rather than externalized step-by-step reasoning traces. However, such models can remain constrained by limited effective computational depth and weak long-horizon credit assignment.

We investigate whether explicitly structuring latent computation around directional subgoals can improve hierarchical reasoning models. Efficient reasoning likely requires multi-timescale computation: fast local transformation steps coupled with slower mechanisms that steer computation over longer horizons and allocate additional compute when needed. HRM operationalizes this via coupled fast/slow recurrent loops with adaptive halting (Wang et al., 2025). In many real-world settings, intelligent behavior involves not only optimizing toward a fixed objective but continuously steering computation as new information arrives. A system that can propose short-horizon internal directions, commit to them briefly, and revise them dynamically may better support adaptive reasoning than one that passively minimizes a single global loss.

We propose **Subgoal-Augmented HRM**, an extension of HRM that integrates a FeUdal-style latent subgoal mechanism directly into hierarchical latent computation. The high-level module periodically emits a normalized directional goal vector representing medium-horizon intent. This subgoal influences computation in two complementary ways: (i) it biases internal updates at one or both hierarchical levels through goal-conditioned additive modulation, and (ii) it defines an intrinsic feudal alignment loss that rewards the low-level trajectory for progressing in the goal direction. Optional gating allows the model to modulate commitment strength when subgoals are uncertain. Importantly, our approach introduces only lightweight intrinsic supervision and does not require explicit symbolic decomposition. Subgoals operate entirely in latent space, shaping computation geometrically rather than through tokenlevel planning.

Contributions. (1) We introduce Subgoal-Augmented HRM, integrating directional latent subgoals into hierarchical latent reasoning. (2) We propose a cosine-based feudal alignment loss that provides lightweight intrinsic supervision across hierarchical levels. (3) We empirically demonstrate consistent gains on ConceptARC/ARC-mini and analyze sensitivity to intrinsic weight and goal update period.

2 Background and Related Work

Chain-of-Thought prompting improves reasoning by externalizing intermediate traces but often increases latency and may require curated demonstrations or careful prompting (Wei et al., 2022). Latent reasoning architectures instead aim to internalize multi-step computation inside hidden representations, reducing reliance on explicit intermediate text while still enabling iterative refinement.

Adaptive computation mechanisms such as Adaptive Computation Time (ACT) allocate variable numbers of

internal steps, increasing effective depth when needed (Graves, 2016). HRM combines multi-timescale recurrence (slow/fast loops) with halting to achieve deep latent computation using compact models trained from scratch (Wang et al., 2025). However, coordination between levels remains implicit, with no explicit medium-horizon intent signal.

Feudal-style hierarchical control introduces an explicit interface between intent and execution: a manager emits latent subgoals and a worker is incentivized to follow them (Vezhnevets et al., 2017). We build on this idea, but apply it to latent hierarchical reasoning rather than RL control: subgoals become geometric constraints that structure internal computation, not external actions.

3 Method

3.1 HRM backbone

We build on HRM with two coupled latent states: a slow high-level state z_t^{H} and a fast low-level state z_t^{L}. Given an encoded input $\tilde{x}_t$, the model performs iterative latent computation within a segment using recurrent transformer blocks. At micro-step t, the low-level state updates every step:

$$z_{t+1}^{\mathrm{L}} = f_{\mathrm{L}}(z_t^{\mathrm{L}}, z_t^{\mathrm{H}}, \tilde{x}_t;\, \theta_{\mathrm{L}}). \tag{1}$$

The high-level state updates every T low-level steps:

$$z_{t+1}^{\mathrm{H}} = \begin{cases} f_{\mathrm{H}}(z_t^{\mathrm{H}}, z_{t+1}^{\mathrm{L}};\, \theta_{\mathrm{H}}), & (t+1) \bmod T = 0, \\ z_t^{\mathrm{H}}, & \text{otherwise.} \end{cases} \tag{2}$$

After N micro-steps, the segment produces a prediction from the final high-level state:

$$\hat{y} = f_{\mathrm{O}}(z_N^{\mathrm{H}};\, \theta_{\mathrm{O}}). \tag{3}$$

HRM training uses deep supervision across segments with truncated gradients by detaching state between segments (Wang et al., 2025). HRM also uses an ACT-style halting objective (Graves, 2016) to choose the number of executed segments per example; we treat this as part of the HRM baseline objective.

3.2 Subgoal-Augmented HRM: directional latent intent

We introduce a FeUdal-style manager–worker interface inside HRM's latent computation. The manager (high-level state) periodically emits a directional subgoal that persists for multiple low-level steps and provides an intrinsic alignment signal.

Subgoal emission (manager period P). Every P low-level micro-steps, at times $t_k = kP$, the high-level state emits a goal direction:

$$\tilde{g}_k = W_g\, z_{t_k}^{\mathrm{H}}, \tag{4}$$

$$g_k = \frac{\tilde{g}_k}{\|\tilde{g}_k\|_2 + \varepsilon}. \tag{5}$$

Optionally, a scalar commitment gate is predicted:

$$\alpha_k = \sigma\big(w_\alpha^\top z_{t_k}^{\mathrm{H}}\big) \in (0, 1). \tag{6}$$

For steps $t \in [t_k, t_k + P)$, the active goal and gate are $g(t) = g_k$ and $\alpha(t) = \alpha_k$.

Why direction (not a target state)? A directional goal encodes "where to move" in latent space without forcing the worker to hit an absolute latent target, which can be brittle under nonstationary internal dynamics.

Subgoal injection (goal-conditioned additive bias). We inject the active goal as an additive bias into the internal computation. Let V_{L} and V_{H} be learned projections into the low/high latent spaces.

Low-level injection (default):

$$z_{t+1}^{\mathrm{L}} = f_{\mathrm{L}}\big(z_t^{\mathrm{L}}, z_t^{\mathrm{H}}, \tilde{x}_t + \alpha(t)\, V_{\mathrm{L}} g(t);\, \theta_{\mathrm{L}}\big). \tag{7}$$

High-level injection (optional, on high-level update steps):

$$z_{t+1}^{\mathrm{H}} = \begin{cases} f_{\mathrm{H}}\big(z_t^{\mathrm{H}}, z_{t+1}^{\mathrm{L}} + \alpha(t)\, V_{\mathrm{H}} g(t);\, \theta_{\mathrm{H}}\big), & (t+1) \bmod T = 0, \\ z_t^{\mathrm{H}}, & \text{otherwise.} \end{cases} \tag{8}$$

Intuitively, this converts medium-horizon intent into a persistent steering term that shapes multiple internal updates, rather than relying only on emergent cross-level interactions.

3.3 Feudal alignment loss (intrinsic geometric constraint)

We encourage the worker's latent trajectory to progress along the emitted direction. Over interval $[t_k, t_k + P)$, define the net low-level displacement:

$$\Delta z_k^{\mathrm{L}} = z_{t_k+P}^{\mathrm{L}} - z_{t_k}^{\mathrm{L}}. \tag{9}$$

We define a cosine-based feudal alignment loss:

$$\mathcal{L}_{\mathrm{feudal}}^{(k)} = 1 - \cos\big(\Delta z_k^{\mathrm{L}}, g_k\big) = 1 - \frac{\langle \Delta z_k^{\mathrm{L}}, g_k \rangle}{\|\Delta z_k^{\mathrm{L}}\|_2\, \|g_k\|_2}. \tag{10}$$

Optionally gated:

$$\mathcal{L}_{\mathrm{feudal}}^{(k)} = \alpha_k \Big(1 - \cos\big(\Delta z_k^{\mathrm{L}}, g_k\big)\Big). \tag{11}$$

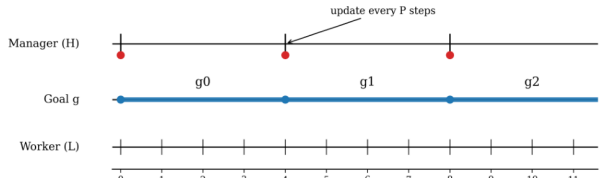

Figure 1: Subgoal update schedule in Subgoal-Augmented HRM. The manager (H) emits a normalized directional latent subgoal every P worker steps; the worker (L) conditions on the current subgoal between updates.

3.4 Full training objective

Let $\mathcal{L}_{\text{HRM}}$ denote the baseline HRM objective including task loss and halting losses (unchanged from HRM (Wang et al., 2025)). We add the intrinsic term weighted by λ (feudal weight):

$$\mathcal{L} = \mathcal{L}_{\text{HRM}} + \lambda \sum_{k} \mathcal{L}^{(k)}_{\text{feudal}}. \tag{12}$$

Because HRM detaches between segments, the feudal loss is computed within segments, consistent with HRM's truncated-gradient training.

4 Experiments

We evaluate Subgoal-Augmented HRM on ARC-style abstract reasoning tasks, focusing on ConceptARC/ARC-mini. These are grid-based visual reasoning puzzles requiring multi-step abstraction and transformation, making them suitable for probing hierarchical latent reasoning and temporal abstraction.

4.1 Dataset and preprocessing

Our primary training corpus is derived from the ARC-AGI and ConceptARC datasets, using the `arc-aug-1000` configuration, which combines original puzzles with extensive data augmentation. Each puzzle is represented as a flattened 30×30 grid sequence with a vocabulary of 12 symbols (padding, end-of-sequence, and colors). We apply several augmentations during training: (i) dihedral transformations (all rotations and reflections), (ii) random color permutations, and (iii) translational padding-based shifts. Each puzzle is augmented up to 1000 times, following standard ARC augmentation practices. For certain analyses and additional validation, we also report results on ConceptARC-mini, consisting of approximately 4,900 training samples and 400 validation/test samples grouped by puzzle type.

Table 1: Baseline comparison on ConceptARC/ARC-mini.

Configuration	LM Loss ↓	Accuracy ↑
HRM baseline ($\lambda = 0.0$)	1.6396	0.6588
Subgoal HRM ($\lambda = 0.05,\ P = 4$)	1.5686	0.7104

4.2 Model configuration and training setup

All experiments use the same HRM backbone architecture, differing only in the presence and hyperparameters of the subgoal mechanism.

HRM backbone. Hidden size 512; high-level layers 4; low-level layers 4; attention heads 8; recurrent cycles $H_{\text{cycles}} = 2$, $L_{\text{cycles}} = 2$; ACT with a maximum of 16 internal steps. We employ RoPE, transformer blocks with self-attention, SwiGLU feedforward layers, and RMS normalization.

Subgoal mechanism. The subgoal head projects z^{H} to a 512-d goal vector, normalized to unit length. A learned scalar gate optionally modulates subgoal injection strength and/or the intrinsic alignment loss. Subgoals update every manager period P.

Optimization. AdamATan2 optimizer, base learning rate 10^{-4} and weight decay 0.1. Puzzle-specific embeddings use learning rate 10^{-2}. We apply a cosine schedule with 2000 warmup steps. Global batch size is 768. All runs are trained for sufficiently long horizons to ensure convergence, with periodic evaluation and multiple random seeds for replication.

4.3 Evaluation metrics

We track: LM loss (token-level stablemax cross-entropy, masked over valid outputs), token-level accuracy, exact accuracy (all tokens correct), feudal loss, and average ACT halting depth. Primary comparisons focus on LM loss and accuracy.

5 Results

5.1 Baseline comparison: effect of latent subgoals

We compare the original HRM model without intrinsic supervision ($\lambda = 0.0$) to Subgoal-Augmented HRM with directional subgoal enabled. Using $\lambda = 0.05$, $P = 4$, we observe consistent improvements. Introducing latent subgoals reduces LM loss by $\sim$4.3% and improves accuracy substantially. The gains are modest in absolute terms but notable given the lightweight nature of the modification: no additional external supervision is introduced, and the backbone remains compact.

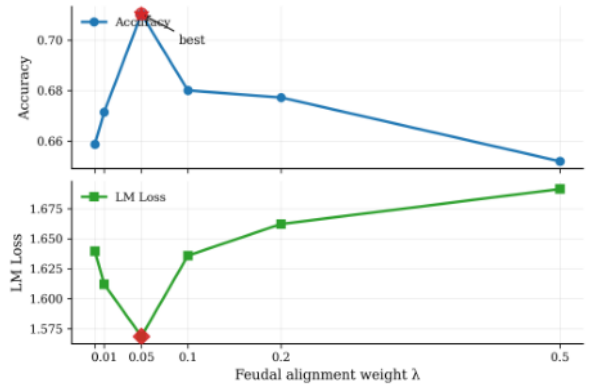

Figure 2: Effect of feudal alignment weight λ on ConceptARC/ARC-mini. A moderate λ improves accuracy and LM loss, while larger values degrade performance.

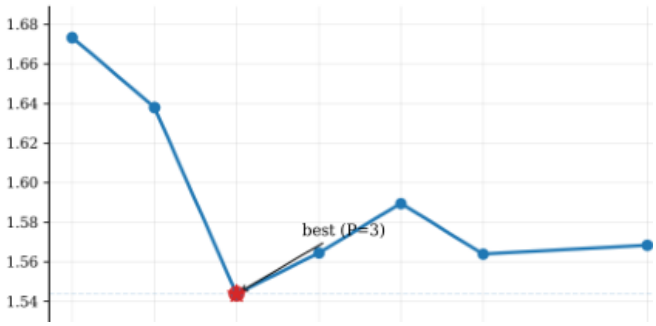

Figure 3: Manager period sweep P (subgoal update steps), fixed $\lambda = 0.05$ (ConceptARC/ARC-mini). Best LM loss at $P = 3$; very small or large P degrades.

5.2 Effect of feudal alignment weight λ

We sweep λ while holding manager period fixed at $P = 4$. Performance peaks at $\lambda = 0.05$. Larger weights degrade performance, consistent with the feudal term acting best as a gentle planning prior rather than a dominant constraint.

5.3 Effect of subgoal update frequency (manager period P)

We sweep P while fixing $\lambda = 0.05$. Moderate periods $P \in [3, 6]$ consistently perform best, while $P = 1$ performs worst. This supports a stability–adaptivity tradeoff: overly frequent re-planning injects high-variance steering signals that destabilize latent computation, whereas moderately persistent goals provide consistent medium-horizon guidance.

Table 2: Feudal weight sweep at fixed $P = 4$.

λ	LM Loss ↓	Accuracy ↑
0.00	1.6396	0.6588
0.01	1.6121	0.6715
0.05	1.5686	0.7104
0.10	1.6359	0.6801
0.20	1.6622	0.6773
0.50	1.6914	0.6520

Table 3: Manager period sweep at fixed $\lambda = 0.05$ (ConceptARC/ARC-mini), reporting LM loss.

P	LM Loss ↓
1	1.6735
2	1.6381
3	1.5438
4	1.5639
5	1.5896
6	1.5644
8	1.5684

5.4 Replication and reproducibility

To assess robustness, we replicate best configurations across seeds. For $\lambda = 0.05$, $P = 3$ (5 runs): mean LM loss 1.5946 (std 0.0454), min 1.5396, max 1.6624. For $\lambda = 0.05$, $P = 4$ (5 runs): mean LM loss 1.6013 (std 0.0475). Both configurations show consistent convergence with low variance.

5.5 Generalization across tasks (ConceptARC-mini)

On ConceptARC-mini, HRM baseline achieves LM loss 2.3163, while Subgoal-Augmented HRM at ($\lambda = 0.05$, $P = 3$) achieves LM loss 2.3079. The improvement is small ($\sim$0.4%) but directionally consistent.

5.6 Ablations

We evaluate ablations removing goal normalization and gating. Both changes reduce performance relative to the default mechanism, supporting the importance of directional semantics (unit-vector goals) and controlled commitment strength when the subgoal signal is uncertain.

6 Analysis and Discussion

The results indicate that explicitly introducing medium-horizon latent subgoals can strengthen hierarchical latent

reasoning, but only in a balanced regime. Subgoals act as a geometric planning prior. In HRM, cross-level coordination emerges implicitly via recurrent state exchange. Subgoal-Augmented HRM makes the interface explicit: the high-level module proposes a medium-horizon direction and injects it persistently as an additive bias, while the feudal alignment term rewards the worker for making net progress along that direction. This provides a local signal connecting short-horizon updates to longer-horizon intent, reducing reliance on purely emergent long-horizon credit assignment.

Why λ has a narrow sweet spot. When λ is too small, the intrinsic signal is too weak to reliably shape latent dynamics. When λ is too large, the intrinsic geometric constraint can conflict with task optimization, reducing representational flexibility. This is consistent with auxiliary objectives being most effective when they softly regularize learning rather than dominate.

Stability–adaptivity tradeoff in goal update frequency. Frequent re-planning (e.g., $P = 1$) creates rapidly changing directions that prevent coherent multi-step internal trajectories. Moderate persistence provides stability long enough for meaningful medium-horizon shaping, while very long persistence risks staleness as internal states evolve.

7 Limitations and Conclusion

Limitations. While Subgoal-Augmented HRM yields consistent improvements, the gains are modest, indicating that directional subgoals act as a lightweight planning prior rather than a large architectural shift. Benefits appear in a narrow regime ($\lambda \approx 0.05$, $P \approx 3$–6), and performance degrades when goals are updated too frequently (e.g., $P = 1$), suggesting sensitivity to stability–adaptivity tradeoffs. Evaluation is primarily restricted to ARC-style puzzle tasks; broader validation is needed. Finally, the subgoal mechanism is intentionally simple; richer goal parameterizations or adaptive manager periods may yield stronger gains.

Conclusion. We introduced Subgoal-Augmented HRM, a feudal-style extension of HRM that makes hierarchical coordination explicit via directional latent subgoals. The high-level module periodically emits a normalized goal direction that is injected into internal computation, and a cosine-based intrinsic alignment loss encourages the worker's latent trajectory to progress along that direction. Across sweeps and replications on ConceptARC/ARC-mini, we find that explicit medium-horizon intent improves performance within a narrow but repeatable regime, and that subgoal persistence is crucial: moderately persistent goals help, while overly frequent re-planning harms performance, consistent with a stability–adaptivity tradeoff.

Impact Statement

This paper proposes a lightweight mechanism for structuring latent computation in compact reasoning models. If such mechanisms improve data/compute efficiency, they could reduce the energy and infrastructure required to train and deploy reasoning systems. Potential risks include misuse in automated decision pipelines or over-reliance on benchmarks that do not reflect real-world requirements; we therefore emphasize evaluation beyond ARC-style tasks and transparent reporting of limitations.

References

[1] François Chollet. On the Measure of Intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[2] François Chollet et al. ARC Prize 2024: Technical Report. *arXiv preprint arXiv:2412.04604*, 2024.

[3] François Chollet et al. ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems. *arXiv preprint arXiv:2505.11831*, 2025.

[4] François Chollet et al. ARC Prize 2025: Technical Report. *arXiv preprint arXiv:2601.10904*, 2026.

[5] DeepSeek-AI. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437*, 2024.

[6] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

[7] Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*, 2016.

[8] Daniel Kahneman. *Thinking, Fast and Slow.* Farrar, Straus and Giroux, 2011.

[9] Alexander Moskvichev, Victor V. Odouard, and Melanie Mitchell. The ConceptARC Benchmark: Evaluating Understanding and Generalization in the ARC Domain. *arXiv preprint arXiv:2305.07141*, 2023.

[10] OpenAI. Introducing OpenAI o1. Online release, 2024.

[11] OpenAI. OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720*, 2024.

[12] Sapient Inc. sapientinc/HRM (GitHub repository). Accessed January 28, 2026.

[13] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

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

[15] Alexander S. Vezhnevets et al. FeUdal Networks for Hierarchical Reinforcement Learning. *arXiv preprint arXiv:1703.01161*, 2017.

[16] Guan Wang et al. Hierarchical Reasoning Model. *arXiv preprint arXiv:2506.21734*, 2025.

[17] Jason Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903*, 2022.