REWARD MOVES WHAT IT MEASURES: ANISOTROPIC PROGRAM ANCHORING FOR RL FINE-TUNING UNDER INCOMPLETE OBJECTIVES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning fine-tuning optimizes generative policies against objectives that are incomplete: a reward scores what it can measure, never everything the designer wants, and optimization then degrades the unmeasured behavior. The standard remedy is a scalar KL penalty toward a frozen clone of the initial policy. This remedy has two limitations: it holds every behavioral dimension equally, so it can only slow the whole policy, and its reference is a snapshot that loses validity once the policy leaves the states it was fitted on. We propose Anisotropic Program Anchoring (APA), which anchors the policy to an executable symbolic program instead. The program encodes the unmeasured behavior directly and can be queried at any state the policy reaches, and one anchor coefficient per behavioral dimension lets the reward move what it measures while the program holds the rest. We evaluate APA on a musical task where the objective is incomplete by nature: a piano and a bass agent fine-tuned together to accompany a jazz soloist. In a factorial design, we compare ten experimental conditions on held-out measures of harmonic competence that the reward never scores. Without anchoring, competence falls by half. APA keeps competence within 1% of its initial value while realizing 80% of the unanchored reward gain, against about 55% for scalar anchors. Under distribution shift it loses 4% of competence where an anchor to a frozen clone loses 38%. Adding another reward term for the most visible failure repairs only that property, while others keep degrading.

1 INTRODUCTION

Automatic music generation has sustained two traditions since the 1950s. A rule-based symbolic generator encodes what the music *is*: a theorist or composer writes the procedure, and the system's competence is whatever those rules express, not only textbook regularities, but the idiosyncratic devices an individual composer invents. Or, a constraint- or search-based system encodes what the music must *satisfy*: the designer states properties and a solver returns objects meeting them (Anders & Miranda, 2011). Both lineages have run alongside each other for the field's entire history (Fernández & Vico, 2013). Deep generative models did not end the split. They learn a corpus's distribution and sample from it, but the generated content tends to mimic the training set rather than exhibit real creativity, and often lack direct means of fine-controlling results on musical constraints the designer has in mind (Briot et al., 2020). The older traditions persist because each supplies what the other lacks: an identity with no objective, and an objective with no identity.

We read this persistence as evidence about objectives in general, not only about music. Reinforcement learning (RL) fine-tuning moves a pretrained generative policy toward behavior its corpus does not supply by replacing the corpus with a reward. A reward, however, is a partial specification: it scores what it can measure and is silent on the rest. Because behavioral dimensions share a network, optimizing the reward comes at a cost: the policy gives up unscored behavior to gain reward. In music accompaniment, a reward can easily compute whether the piano's highest pitch sits just below the soloist's lowest, but no formal reward covers everything a listener judges about a voicing. Adding a term for the property that visibly failed repairs that property, while others keep degrading.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The standard alternative is to restrain the policy: a scalar KL penalty toward a frozen copy of the policy before fine-tuning (Jaques et al., 2017; Ouyang et al., 2022). This penalty has two limitations. It is *isotropic*: one coefficient holds every dimension equally, including those the reward is meant to change, so it can only make a policy more conservative everywhere. And its reference is a *snapshot*, accurate only on the states it was fitted on, which fine-tuning leaves.

We propose **Anisotropic Program Anchoring** (APA), which addresses both limitations and combines the two traditions: the program supplies the identity, the reward the objective. The reference becomes a rule-based generator program. It is written from domain knowledge rather than fitted to data, it can be read and edited in musical terms, and because it can be executed during training, it gives an answer at every state the policy reaches, including states no training data covered. The anchor strength is a vector with one coefficient β_k per behavioral dimension, which the program's modules declare as separate outputs. Setting $\beta_k = 0$ on the reward's target dimensions and $\beta_k > 0$ on all others lets the reward lead a search toward novel behavior while the program holds the rest. The designer controls the result in two places, the rules of the program and the terms of the reward.

Following a divide-and-conquer strategy, hand-written generative music systems are commonly decomposed into interacting, role-specialized agents (Tatar & Pasquier, 2019). We therefore test APA with several agents fine-tuned at once. In our testbed, a piano and a bass agent accompany a fixed jazz soloist agent under a reward that scores only where their pitch ranges sit relative to each other, while the harmonic correctness of what they play is left to the anchor.

Contributions.

- We propose APA, an anchor to a symbolic program with one coefficient per declared behavioral dimension (Section 3). The program only has to return actions as the anchor is designed through cross-entropy, which in expectation minimizes forward KL to the program's action distribution.
- The standard KL penalty implicitly combines a scalar coefficient with a frozen reference. We separate the two choices, shape (scalar or anisotropic) and reference (frozen clone or live program), and cross them in a factorial experimental design (Section 4).
- We introduce a multi-agent jazz accompaniment testbed with an incomplete reward and held-out measures of harmonic competence (Section 4).
- At $\beta = 0.3$, anisotropic anchors keep competence within 1% of the clone while realizing 80% of the unanchored reward gain, against about 55% for scalar anchors (Section 5.1). Under distribution shift, the program-anchored policy loses 4% of its competence and the clone-anchored one 38% (Section 5.2). Adding a reward term for the most visible failure repairs only that channel.

2 Related Work

Reward misspecification. Optimizing an incomplete reward degrades the properties it does not reference (Amodei et al., 2016; Zhuang & Hadfield-Menell, 2020), and the degradation grows with optimization pressure (Pan et al., 2022; Gao et al., 2023); Karwowski et al. (2024) derive early-stopping rules for this regime. Skalse et al. (2022) show that non-trivial unhackable reward pairs exist only when the set of policies is restricted, and Baker et al. (2025) find that penalizing a detected failure can teach agents to conceal it. Instead of completing the reward, we restrict the policy, and we test reward completion as a baseline (Section 5.4).

KL regularization toward a reference policy. A KL penalty toward the policy before fine-tuning was introduced for sequence models by Jaques et al. (2017) and is standard in RLHF (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022). Variants change the reference, for example to a weight-averaged moving anchor (Ramé et al., 2024) or to several reference models (Le et al., 2025), or change the divergence, for example to a χ^2 divergence between occupancy measures (Laidlaw et al., 2025); Liu et al. (2025) analyze the role of the KL term in the gradient. Shenfeld et al. (2026) show that KL to the base policy predicts forgetting, and Luo et al. (2026) that a KL evaluated on the current state distribution does not detect drift elsewhere. In music accompaniment, Wu et al. (2024) anchor to a separately trained model because anchoring to the initialization failed, and Wu et al. (2026) add adversarial post-training because a KL penalty did not prevent reward hacking. In all of these the reference is a fitted model and the coefficient a scalar.

Table 1: Prior work by the two properties APA changes.

	Scalar (isotropic)	**Non-uniform (anisotropic)**
Fitted or frozen reference	Jaques et al. (2017); RLHF (Ziegler et al., 2019; Ouyang et al., 2022); Ramé et al. (2024); Laidlaw et al. (2025); Wu et al. (2026)	Archer (Wang et al., 2025a), WPR (Na et al., 2026) *(tokens, embed.)*; EWC (Kirkpatrick et al., 2017), L2-SP (Li et al., 2018) *(param.)*
Executable program	PROPEL (Verma et al., 2019), CORE-RL (Cheng et al., 2019), RUBICON (Liu et al., 2023), PPF (Jung et al., 2025)	**APA (ours)** *(declared behavioral dimensions)*

Non-uniform regularization. Other methods constrain the policy unequally. Archer (Wang et al., 2025a) weights the KL term differently for high- and low-entropy tokens, WPR (Na et al., 2026) replaces it with a Wasserstein distance over a semantic embedding, MPO (Abdolmaleki et al., 2018) bounds the mean and covariance of a Gaussian policy separately, MO-MPO (Abdolmaleki et al., 2020) gives each objective its own KL bound, and Galashov et al. (2019) restrict what the default policy observes. In parameter space, EWC (Kirkpatrick et al., 2017) and L2-SP (Li et al., 2018) weight each parameter's distance to a pretrained model, natural-gradient and trust-region methods use a non-Euclidean metric (Kakade, 2001; Schulman et al., 2015), and Nayak et al. (2026) restrict updates to spectral subspaces. In these methods the axes are derived from entropy, embeddings, curvature or the set of objectives; in APA they are the declared behavioral dimensions of a module, and their coefficients are set in advance.Coefficients per dimension resemble multi-task loss weighting (Kendall et al., 2018; Chen et al., 2018), but those weights balance objectives that are all optimized, whereas ours weight anchors.

Executable priors. Several methods regularize a neural policy toward a hand-written program or controller, including PROPEL (Verma et al., 2018; 2019), CORE-RL (Cheng et al., 2019), RUBICON (Liu et al., 2023), PPF (Jung et al., 2025) and a symbolic regularizer (Murari et al., 2026) for PPO (Schulman et al., 2017). They apply one coefficient to the whole policy, and several anneal it, because the prior serves interpretability, safety or sample efficiency rather than the preservation of unrewarded behavior. Shielding (Alshiekh et al., 2018; Yang et al., 2023) removes actions that a specification forbids, which covers per-action properties but not distributional ones such as the variety of the program's choices. Residual RL (Silver et al., 2018; Johannink et al., 2019) learns a correction on top of a fixed prior, and Wang et al. (2025b) show that under stated assumptions it subsumes KL-regularized RL. Because the program is executed at every state the policy visits, our anchor also queries an expert on the learner's state distribution, as in DAgger (Ross et al., 2011). All these methods target control tasks such as driving, building control and locomotion. In generative RL, hand-written rules have so far entered the reward, as in the music-theory rewards of Jaques et al. (2017), while the anchor remains a fitted model; we use the rule-based generator as an anchor.

3 METHOD

3.1 SETTING

Agents, states and factored actions. One or more agents interact with an environment in discrete time steps t; in our testbed one step is one beat of music. Within a step the agents act one after another in a fixed order. When its turn comes, an agent observes a state s_t and selects an action a_t. The state contains the current context (in our testbed, the chord and the soloist), the actions already taken in this step by agents earlier in the order, and the previous step's actions of agents later in the order. Every agent is therefore part of the others' environment. The action consists of K components, $a = (a^1, \dots, a^K)$, where each component a^k takes values in its own finite set $\mathcal{A}^k$. We call each component a *behavioral dimension*. For a pianist, for example, the pitch class, the octave and the volume of the top note are three separate dimensions.

Policy. A policy π_θ is a neural network with parameters θ that maps a state to one categorical distribution per behavioral dimension, which are sampled independently given the state:

$$\pi_\theta(a \mid s) \;=\; \prod_{k=1}^{K} \pi_\theta^k(a^k \mid s). \tag{1}$$

The network consists of shared layers, the *backbone*, which compute one internal representation of the state, followed by one output layer per dimension, the *head*, which turns that representation into the distribution $\pi_\theta^k(\cdot \mid s)$. Because all heads of one agent read the same backbone, a gradient step that changes one dimension in general changes the others as well.

Live symbolic program. Each agent comes with a hand-written symbolic *program* π^{sym}: a set of rules that computes an action from the state, with the same K dimensions as the policy. Rules may be stochastic, for instance by choosing at random among several notes that fit the current chord. We write $\pi_k^{\mathrm{sym}}(\cdot \mid s)$ for the distribution of dimension k that the program's rules induce at state s. The method assumes only that the program can be executed on any state and returns one sampled action; it never needs π_k^{sym} itself.

Clone and reward. Fine-tuning starts from a *clone*: a policy fitted by behavioral cloning to state–action pairs recorded while the program acts in the environment. We denote the clone by $\hat{\pi}^0$ and keep a frozen copy of it. The policy is then fine-tuned with PPO to maximize the expected value of a reward $r(s, a)$. The reward is incomplete in the sense of Section 1: it judges some properties of the action and is silent on others. We call the dimensions that the designer intends the reward to change its *target dimensions*.

Implementation. Our implementation is based on a modular framework for interactive music systems in which every module declares the typed data it reads and writes (Ostermann, 2026). We further extended its functionalities for automatic policy network construction, cloning and fine-tuning (Ostermann et al., 2026) with APA. A module's declared outputs are its behavioral dimensions, so the factorization in Equation 1 comes from the program's interface. The update step of APA is described in Appendix B, Algorithm 1.

3.2 Anchored fine-tuning

An *anchor* adds a penalty for moving away from a reference policy π^{ref} to the fine-tuning loss. With a separate anchor strength $\beta_k \geq 0$ for each dimension, the loss minimized in each PPO update is

$$\mathcal{L}(\theta) \;=\; \frac{1}{K}\sum_{k=1}^{K}\Big[\,\mathcal{L}_{\mathrm{PPO}}^k(\theta) \;-\; \alpha\,\mathbb{H}\big(\pi_\theta^k\big) \;+\; \beta_k\,\mathcal{A}_k(\theta)\,\Big]. \tag{2}$$

Here $\mathcal{L}_{\mathrm{PPO}}^k$ is PPO's clipped surrogate loss evaluated with the importance ratio of head k alone, $\pi_\theta^k(a^k \mid s)/\pi_{\theta_{\mathrm{old}}}^k(a^k \mid s)$; the entropy term, weighted by α, keeps each head from becoming deterministic too early; and $\mathcal{A}_k$, the *anchor term* of dimension k, measures how far head k has moved from the reference. All terms are averaged over the states visited during the rollout. Equation 2 leaves two choices open for $\mathcal{A}_k$.

Shape of the anchor strength. A *scalar* anchor sets $\beta_k = \beta$ for every k. It pulls every dimension back toward the reference with the same strength, which is why we call it *isotropic*. Raising β reduces how far the policy moves in all dimensions at once, including the target dimensions. An *anisotropic* anchor sets $\beta_k = 0$ on the target dimensions and $\beta_k = \beta$ on all others. It is *anisotropic*: it holds the dimensions the reward is not meant to change and leaves the target dimensions free. Both shapes have one free parameter, β.

Reference. The reference is either the frozen clone, $\pi^{\mathrm{ref}} = \hat{\pi}^0$, or the symbolic program, $\pi^{\mathrm{ref}} = \pi^{\mathrm{sym}}$. With the clone, both distributions are network outputs, so the anchor term is computed exactly per state in either direction, $\mathcal{A}_k = D_{\mathrm{KL}}(\hat{\pi}_k^0 \,\|\, \pi_\theta^k)$ or $\mathcal{A}_k = D_{\mathrm{KL}}(\pi_\theta^k \,\|\, \hat{\pi}_k^0)$. A scalar anchor to the frozen clone in the second direction is the standard penalty of RLHF (Jaques et al., 2017; Ouyang et al., 2022). APA is the combination of a anisotropic anchor with the program as reference.

3.3 Anchoring to a symbolic program

At every step of a rollout, the program is executed on the same state s_t as the learning policy, and its sampled action $\tilde{a}_t$ is recorded. The environment then advances with the policy's action a_t, not the program's. The recorded $\tilde{a}_t$ therefore serves as a label at exactly the states the policy visits, including states that never occurred while the clone's training data was recorded. As in DAgger (Ross et al., 2011), the expert is queried on the learner's own state distribution, at the cost of one program execution per step.

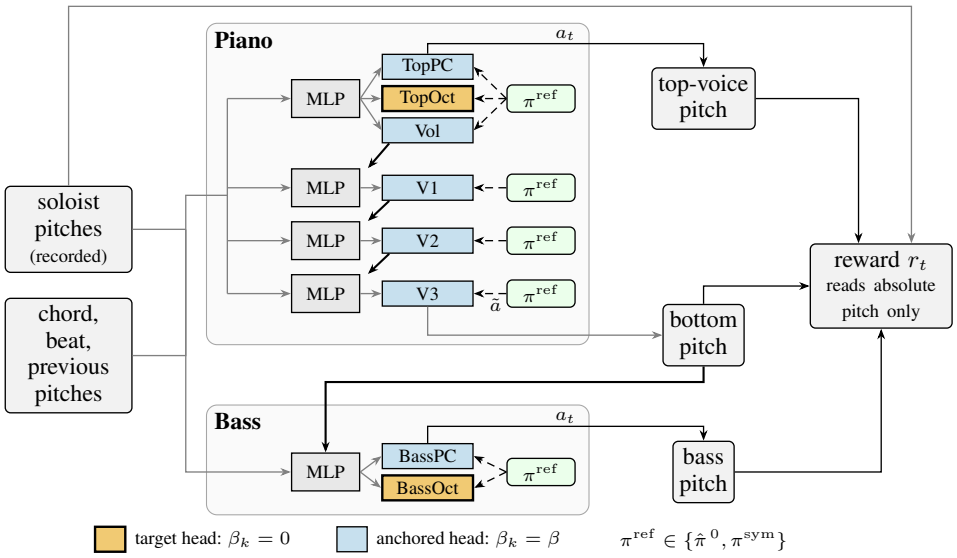


Figure 1: The testbed: A jazz soloist being accompanied by piano (4-note chords) and bass.

The anchor term is the negative log-probability that head k assigns to the program's sampled action, $\mathcal{A}_k = -\log \pi_\theta^k(\tilde{a}_t^k \mid s_t)$, averaged over the rollout. Its expectation over the program's random choices is

$$\mathbb{E}_{\tilde{a}^k \sim \pi_k^{\text{sym}}}\big[-\log \pi_\theta^k(\tilde{a}^k \mid s)\big] \;=\; \mathrm{CE}\big(\pi_k^{\text{sym}}, \pi_\theta^k\big) \;=\; \mathbb{H}\big(\pi_k^{\text{sym}}\big) + D_{\mathrm{KL}}\big(\pi_k^{\text{sym}} \,\|\, \pi_\theta^k\big). \tag{3}$$

The entropy $\mathbb{H}(\pi_k^{\text{sym}})$ does not depend on θ, so in expectation the anchor term has the gradient of the forward divergence $D_{\mathrm{KL}}(\pi_k^{\text{sym}} \,\|\, \pi_\theta^k)$. This divergence is large when the policy assigns low probability to an action the program uses, so it holds the policy to the program's whole distribution rather than to its most frequent action (Ghasemipour et al., 2020). One sample per step suffices, which matters in practice: designers write rules that make decisions, not rules that report probabilities. The reverse direction is not available, because it would require the program's probability of the policy's own actions. Ostermann et al. (2026) offered anchoring a single module to its program's output with a scalar coefficient; here the anchor is set per dimension, read as a divergence to the program's distribution, and compared with a frozen reference. A proposition on the theoretical benefits of the anisotropic anchor is given in Appendix A.

4 Experimental Setup

Testbed. The task is a piano and a bass accompanying a jazz soloist. The soloist is replayed from transcriptions of recorded improvisations in the Weimar Jazz Database (Pfleiderer et al., 2017), restricted to 405 tunes in 4/4 time, and does not react to the other players. Piano and bass are the learning agents and act once per beat. Figure 1 shows one step of the RL environment. The top-voice module reads the chord, the beat, the soloist's line and its own recent notes; each inner-voice module reads the chord and the voices chosen before it; the bass reads the chord, the beat, its own recent notes and the piano's bottom note. A note's pitch can be split into its *pitch class*, one of the twelve note names in $\{\mathrm{C}, \mathrm{C}\sharp, \mathrm{D}, \ldots, \mathrm{B}\}$, and its *octave*, which of the repeated ranges of twelve it lies in and thus how high or low it sounds, its *register*; harmony depends on pitch classes only. A *chord tone* is a pitch class that belongs to the current chord, and a *voicing* is the arrangement of chord tones across the piano's register: which tones are played, in which vertical order, and how far apart. The piano's action has six dimensions: the pitch class (TopPC), octave (TopOct) and volume (Vol) of its top voice, and the pitch classes of its three inner voices (V1–V3). The bass has pitch class (BassPC) and octave (BassOct). Splitting the voicing into four voices gives it several dimensions that can be anchored separately. Each symbolic program derives the admissible notes from the current chord and samples from them. Policies are initialized by behavioral cloning of these programs. For more details on the head shapes and the rule-based behavior of the symbolic agents, see Appendix C.1.

Table 2: The ten conditions (c*) used in the experiments.

Reference	Divergence	Init.	No anchor	Scalar β	Aniso. β_k
none	—	warm	c1	—	—
none, lower lr / clip range	—	warm	c2	—	—
frozen clone $\hat{\pi}^0$	$D_{\mathrm{KL}}(\hat{\pi}_k^0 \,\|\, \pi_\theta^k)$	warm	—	$\mathrm{c3}_{\mathrm{fwd}}$	$\mathrm{c8}_{\mathrm{fwd}}$
frozen clone $\hat{\pi}^0$	$D_{\mathrm{KL}}(\pi_\theta^k \,\|\, \hat{\pi}_k^0)$	warm	—	$\mathrm{c3}_{\mathrm{rev}}$	$\mathrm{c8}_{\mathrm{rev}}$
symbolic program π^{sym}	Eq. 3	warm	—	c4	**c5 (APA)**
symbolic program π^{sym}	Eq. 3	cold	—	c7	c6

Joint reward. The joint reward is the mean of two terms that depend only on absolute pitch. The first is highest when the piano's top voice sits just below the soloist's lowest sounding note; the second is highest when the bass sits about an octave below the piano's bottom voice. Both are triangular functions of the pitch distance (Appendix C.3), so they also reward partial progress toward the target. This simple arranging rule keeps the instruments in separate pitch ranges and stands in as a proxy for criteria that are harder to formalize. The reward says nothing about whether a note belongs to the chord, how the voicing is spaced, or where accents fall. The two octave heads are therefore the target dimensions, and the anisotropic anchor sets $\beta_k = 0$ on TopOct and BassOct and $\beta_k = \beta$ on the other six heads. The pitch-class and inner-voice heads are anchored although the reward depends on them through absolute pitch: register can be changed through the free octave heads, and the reward does not judge which chord tone a voice plays. The clone obtains a joint reward of 0.40 out of a maximum of 1.

Conditions. Table 2 lists the ten test conditions (labeled as c*). *Warm* conditions start from the clone; *cold* conditions start from a randomly initialized network, which tests whether the anchor or the warm start carries the effect (without cloning there is no clone to freeze). Condition 1 has no anchor and is reported as a trajectory through its checkpoints, so comparisons are made against every point the unanchored policy passes through, which also covers early stopping. Condition 2 has no anchor but lowers the learning rate and tightens the clipping range in six combinations (Appendix C.4), to test whether smaller updates have the same effect as an anchor. The complete protocol of the experiment runs is provided in Appendix C.6.

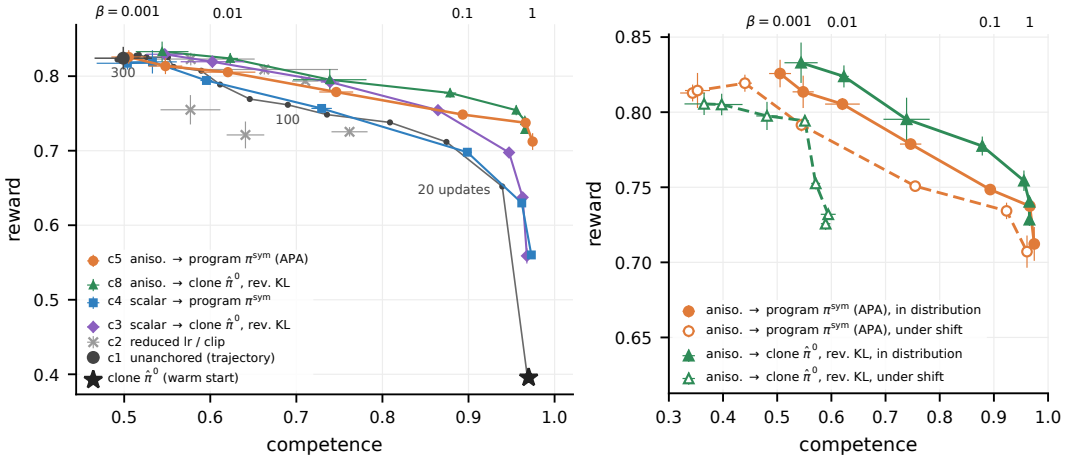
Competence index. Six properties that the reward does not judge are computed at every update. None enters any objective or any selection rule: whether the voicing contains every chord tone (completeness), whether the inner voices, the top voice and the bass play chord tones (three legality rates), whether adjacent voices are three to five semitones apart (spacing), and whether accents fall on downbeats (volume meter). The first four summarize harmonic correctness and their mean builds the *competence* index that falls as harmonic validity and integrity is lost. The main results plot joint reward against competence. Spacing and meter are left out of it because spacing can be changed indirectly through the reward, which reads the piano's bottom note, and the volume head is nearly deterministic under the program; however, both rank the conditions the same way as competence (Appendix D.2). To see which dimensions moved, we report each head's *drift* from the program, $\mathcal{D}_k = \mathrm{CE}(\pi_k^{\mathrm{sym}}, \pi_\theta^k) - \mathbb{H}(\pi_k^{\mathrm{sym}}) = D_{\mathrm{KL}}(\pi_k^{\mathrm{sym}} \,\|\, \pi_\theta^k)$, averaged over visited states, for every condition. It is zero when head k reproduces the program's distribution.

Out-of-distribution shift. To test the two references under shift, a second clone is fitted only on tunes whose chords are dominant- or minor-seventh chords on the six most frequent roots, 39 of the 405 tunes. Fine-tuning then runs on the remaining tunes, which contain chord types and roots this clone never saw (Appendix C.2). The programs derive the admissible pitch classes from any chord and are therefore correct on these chords. We split by chord because the measured competence depends on the chord, not on the soloist.

5 Results

5.1 Anisotropic anchoring dominates at high competence

Figure 2a shows reward against competence for all conditions. At $\beta = 0.3$, where every anchored condition retains competence between 0.955 and 0.966, the anisotropic conditions obtain about



(a) in distribution (b) under shift

Figure 2: Joint reward against harmonic competence. One marker per β. c1 drawn as the trajectory through its checkpoints. Error bars are 95% CI.

0.11 more reward than the scalar conditions with the same reference. This difference amounts to about a quarter of the reward gain that the unanchored condition achieves over the clone, at equal competence and equal training budget. Comparing the frontiers at equal competence gives the same ordering, and the difference grows with the level of competence.

The reward gain depends on how much competence must be retained. Below a competence of about 0.9, the scalar anchor to the frozen clone under reverse KL (standard in RLHF) obtains as much reward as APA, and the anisotropic anchor to the frozen clone obtains the most. Reverse KL is mode-seeking: against a fixed reference, the policy can concentrate on part of the clone's behavior and keep that part at low cost. Above 0.9, the scalar frontiers fall steeply, because a scalar anchor cannot hold the pitch-class heads without also holding the octave heads, whereas both anisotropic frontiers stay more flat. Among the scalar conditions, reverse KL is consistently the stronger baseline , whereas forward KL anchoring to a clone and the symbolic program perform the same (Appendix D.1, Figure 7a). We therefore compare against the stronger configuration throughout. The anchor to the program is a cross-entropy against sampled program actions and therefore corresponds to forward KL; reverse KL would require the program's probability of the policy's own actions, which sampling does not provide (Section 3.3). This restriction follows from the sample-only access to the program, not from the anchoring idea itself.

Unanchored fine-tuning degrades competence Without an anchor (c1), fine-tuning roughly doubles the clone's reward, while competence falls to 0.50. The two quantities change on different timescales. Reward is within one percent of its final value by update 200, whereas competence keeps falling after reward has plateaued, from 0.55 at update 200 to 0.50 at update 300 (Appendix C.6, Figure 6). Stopping early therefore selects a different point on the same trajectory, not a better one. For this reason c1 enters every comparison below as its full trajectory of checkpoints rather than as an endpoint.

Slowing learning is not enough. Condition 2 lowers the learning rate and tightens the clipping range in six combinations. No combination exceeds a competence of 0.76, and the combination that comes closest gives up a quarter of the unanchored reward gain. Smaller steps reduce how far all heads move, not which heads move, so they can retain competence only by also reducing movement in the heads the reward is meant to change (Section 2).

The effect does not depend on warm start. From a random initialization, the difference between the anisotropic and the scalar condition remains: at $\beta = 0.3$ they obtain 0.742 and 0.626 reward. The cold anisotropic condition needs a larger β to reach the same competence (0.920 at $\beta = 0.3$, against 0.966 from the warm start), so behavioral cloning speeds up retention but is not the source of the result (Appendix D.3).

Table 3: In-distribution drift per head at $\beta = 0.3$. V1–3 is the mean over the three inner voices; all heads separately in Table 8.

	held by every anchor			free under aniso.		
condition	TopPC	BassPC	V1–3	TopOct	BassOct	reward
c1 unanchored	5.37	5.86	2.76	1.53	1.39	0.824
c3 scalar → clone (rev.)	1.31	0.78	0.45	0.59	0.34	0.637
c4 scalar → program	1.27	0.59	0.36	0.55	0.31	0.630
c8 aniso. → clone (rev.)	1.26	0.80	0.43	1.30	2.38	0.740
c5 **aniso. → program**	1.22	0.59	0.35	1.24	2.36	0.737

5.2 The symbolic program remains valid under distribution shift

In distribution, the choice of reference makes little difference. At $\beta = 0.3$, anisotropic anchoring to the symbolic program and to a frozen clone of it reach the same competence (0.966 and 0.966) and nearly the same reward (0.737 and 0.740), and their per-head drift is similar (Table 3). The same holds for the scalar conditions. With this alone, a frozen imitation of the program would serve as well as the program itself.

This holds only as long as the states the policy reaches resemble those the clone was fitted on. To test this, we fit a second clone on a subset of the music corpus (Section 4), and then fine-tuned on the remainder. The program derives the admissible pitch classes from any chord symbol and is therefore correct on unseen chords, whereas the clone can only handle chords similar to its data. Because the remainder still contain many chords the clone has seen, the visible effect is a lower bound.

Under the shift, the program-anchored condition loses 0.04 of competence at $\beta = 0.3$ and reaches 0.961 at $\beta = 1$, within 0.013 of its value in distribution (Figure 2b). The clone-anchored condition loses 0.37, and increasing the anchor strength does not recover it: its competence stays at or below 0.6 for every β. A stronger anchor to an inaccurate reference holds the policy closer to that reference's errors, so the clone-anchored frontier levels off at a lower ceiling.

The difference is not paid for with reward. Under the shift, both conditions obtain nearly the same reward at $\beta = 0.3$ (about 0.73), so the difference of 0.33 in competence does not come at the cost of task performance. The difference also remains when the clone-anchored sweep is read at exactly the program-anchored condition's reward, whereas the same comparison in distribution shows no difference.

5.3 Which dimensions drift

Table 3 locates the reward difference of Figure 2a. On the pitch-class heads and on the inner voices, all anchored conditions drift similarly, regardless of the reference and KL direction. The conditions differ on the two octave heads: there the anisotropic conditions ($\beta_k = 0$) drift two to eight times further than the scalar conditions. The octave heads are the targets of the reward, and the scalar conditions lose reward by holding them.

Two further observations qualify this result. First, a coefficient of zero does not leave a head entirely unconstrained. The top octave head shares its network with the anchored top pitch class and volume heads, and under the anisotropic anchor it drifts less than without any anchor (1.24–1.30 against 1.53). The bass octave head, whose only neighbor is the anchored bass pitch class, drifts further (2.36–2.38 against 1.39), since, with the pitch class held, the reward can act on the octave alone. Second, the same separation appears in the policy entropy (Appendix D.5). On the octave heads the anisotropic conditions keep about 0.25 (top voice) and 0.5 (bass) of the clone's entropy, against 0.6 to 1.0 for the scalar conditions, with no overlap and no difference on the heads both families hold.

Figure 3 shows the per-head drift from the program in distribution and under shift. The drift of the program-anchored conditions changes by at most 0.08 between the two settings, whereas the drift of the clone-anchored conditions increases by a factor of 2.1 to 39, most strongly on the inner voices. Under shift, the four clone-anchored conditions differ at most by a factor of 0.14 on any head, so the effect is due to the reference and not to the anchor's shape or the direction of its divergence

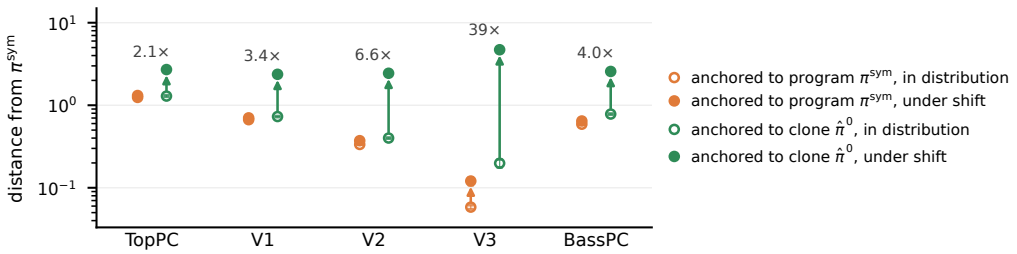


Figure 3: In- and out-of-distribution drift from the program per head at $\beta = 0.3$.

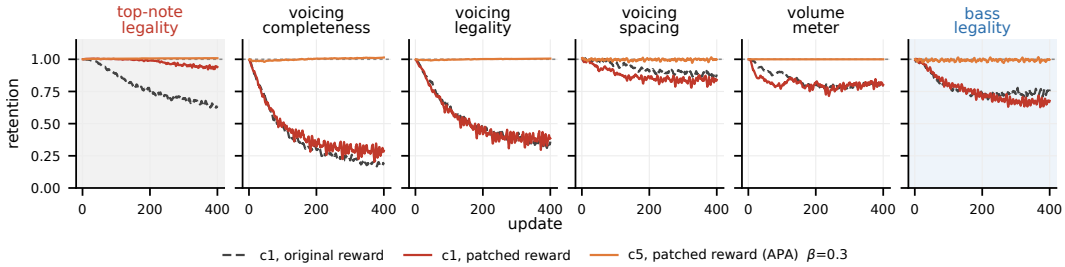


Figure 4: Each held-out channel as a fraction of the clone's value.

(Appendix D.4). A reference fitted to data is inaccurate where the data did not reach, and the policy inherits these errors on the dimensions whose harmonic content the reward does not judge.

5.4 Patching the reward does not prevent degradation

An obvious response to an incomplete objective is to complete it. To test whether completing the reward removes the need for an anchor, a patched reward adds, with equal weight, a term for the piano's top note being a chord tone (Appendix C.3). Figure 4 shows the result: The patch raises top-note legality from 0.64 to 0.94 of the clone's value, but the improvement does not extend to the other channels. Voicing completeness and voicing legality improve slightly and remain far below the clone, spacing and volume meter do not improve, and bass legality even suffers under the patched objective. Adding a reward term thus improves the channel it names, while other channels continue to deteriorate. Completing the objective in this way requires enumerating in advance every property that should be preserved, and turns a single objective into a problem of weighting competing terms (Abdolmaleki et al., 2020) and expensive reward shaping (Ng et al., 1999).

6 Conclusion

We introduced APA, which anchors RL fine-tuning to an executable symbolic program with one coefficient per behavioral dimension. In a multi-agent jazz accompaniment task, anisotropic anchors kept the clone's harmonic competence while realizing 80% of the unanchored reward gain, against about 55% for scalar anchors, and anchoring to the program rather than a frozen clone preserved competence under distribution shift.

Limitations and outlook. APA requires a program that covers every anchored dimension and runs on any state the policy reaches; such programs are common in symbolic music generation but rarer in other domains, and someone has to write them. The coefficients were binary and set by reading the reward, and a coefficient of zero does not free a head entirely, since the heads of one module share a network. Because the program returns only samples, the anchor is limited to forward KL; a clone fitted online to the program's samples would estimate the action probabilities that reverse KL requires. Finally, we kept the testbed intentionally small to isolate the mechanism. Applying APA to richer systems, for instance with rhythm generation, would make a listening study meaningful, and applying it to other domains, such as control tasks like autonomous driving, procedural generation or game-playing agents, would test its generality.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology or experiments, formulate mathematical claims, assist in the writing of proofs, and clean and reformat dataset. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, implement methods, assist with translation, support qualitative and thematic data analysis, and interpret results. Additionally, we used generative AI tools to create or edit software code, create or modify scientific figures or images, draft parts of a research paper, summarize or analyse existing literature, brainstorming, identify relevant literature, and format references. We have reviewed all AI-assisted work. LLM-coding abilities were mainly used to find bugs and test edge cases. Every line of code was manually reviewed by the first author. The same applies to the support in mathematical reasoning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

This work is fully reproducible.
Everything below will be released in a public and permanent repository in case of paper acceptance.

The framework code to replicate the experiments is available from:
`https://nx44933.your-storageshare.de/s/JxPK2869FmwYPTi`

The jazz trio testbed is available from:
`https://nx44933.your-storageshare.de/s/W5DdGsbYTcL99na`

The dataset used can be downloaded from:
`https://jazzomat.hfm-weimar.de/`
(however, it is also already bundled with the testbed, as the dataset is under *Open Data Commons Open Database License (ODbL) v1.0*)

We also provide all the data that we obtained in the experiments and derived our results from:
`https://nx44933.your-storageshare.de/s/Yf4BawYE6LqSoJy`

Finally, we provide some audio files for an impression of the music quality:
`https://nx44933.your-storageshare.de/s/rbTxe37wXtrgBn7`
(please keep in mind that music quality was not a goal of this paper; on the contrary, we kept the jazz trio testbed simple on purpose in order to better isolate the mechanism.)

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

Abbas Abdolmaleki, Sandy Huang, Leonard Hasenclever, Michael Neunert, Francis Song, Martina Zambelli, Murilo Martins, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. A distributional view on multi-objective policy optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11–22. PMLR, 2020.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11797.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Torsten Anders and Eduardo R. Miranda. Constraint programming systems for modeling music theories and composition. *ACM Computing Surveys*, 43(4):30:1–30:38, 2011. doi: 10.1145/1978802.1978809.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.

Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer, Cham, 2020. doi: 10.1007/978-3-319-70163-9.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803. PMLR, 2018.

Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1141–1150. PMLR, 2019.

Jose D. Fernández and Francisco Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013. doi: 10.1613/jair.3908.

Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations (ICLR)*, 2019.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1259–1277. PMLR, 2020.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with KL-control. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1645–1654. PMLR, 2017.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Hyunyoung Jung, Zhaoyuan Gu, Ye Zhao, Hae-Won Park, and Sehoon Ha. PPF: Pre-training and preservative fine-tuning of humanoid locomotion via model-assumption-based regularization. *IEEE Robotics and Automation Letters*, 10(11):11466–11473, 2025. doi: 10.1109/LRA.2025.3608637.

Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, pp. 1531–1538, 2001.

Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar Skalse. Goodhart's law in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. Correlated proxies: A new definition and improved mitigation for reward hacking. In *International Conference on Learning Representations (ICLR)*, 2025.

Hung Le, Quan Hung Tran, Dung Nguyen, Kien Do, Saloni Mittal, Kelechi Ogueji, and Svetha Venkatesh. Multi-reference preference optimization for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24375–24383, 2025. doi: 10.1609/aaai.v39i23.34615.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2825–2834. PMLR, 2018.

Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. Rule-based policy regularization for reinforcement learning-based building control. In *Proceedings of the 14th ACM International Conference on Future Energy Systems (e-Energy '23)*, pp. 242–265. ACM, 2023. doi: 10.1145/3575813.3595202.

Kezhao Liu, Jason Klein Liu, Mingtao Chen, and Yiming Liu. Rethinking KL regularization in RLHF: From value estimation to gradient optimization. *arXiv preprint arXiv:2510.01555*, 2025.

Mao-Lin Luo, Zhe-Xu Wang, Zi-Hao Zhou, Bo Ye, Jian Zhao, Min-Ling Zhang, and Tong Wei. RL forgets! towards continual policy optimization. *arXiv preprint arXiv:2607.04364*, 2026.

Simone Murari, Celeste Veronese, and Daniele Meli. Sample-efficient neuro-symbolic proximal policy optimization. *arXiv preprint arXiv:2604.25534*, 2026.

Byeonghu Na, Hyungho Na, Yeongmin Kim, Suhyeon Jo, HeeSun Bae, Mina Kang, and Il-Chul Moon. Semantic-aware Wasserstein policy regularization for large language model alignment. In *International Conference on Learning Representations (ICLR)*, 2026.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, and Akash Srivastava. Sculpting subspaces: Constrained full fine-tuning in LLMs for continual learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

Fabian Ostermann. BbMuse: A blackboard-driven framework for real-time interactive music. In *Proceedings of the International Computer Music Conference (ICMC)*, pp. 273–280, Hamburg, Germany, 2026.

Fabian Ostermann, Marco Pleines, and Günter Rudolph. BbLearn: Sculpting behavioral clones in modular blackboard-driven music systems. In *Proceedings of the 7th Conference on AI Music Creativity (AIMC)*, Berlin, Germany, 2026. doi: 10.5281/zenodo.22255603.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations (ICLR)*, 2022.

Martin Pfleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart (eds.). *Inside the Jazzomat. New Perspectives for Jazz Research*. Schott Campus, Mainz, 2017.

Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, and Olivier Bachem. WARP: On the benefits of weight averaged rewarded policies. *arXiv preprint arXiv:2406.16768*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations (ICLR)*, 2026.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9460–9471, 2022.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021, 2020.

Kıvanç Tatar and Philippe Pasquier. Musical agents: A typology and state of the art towards musical metacreation. *Journal of New Music Research*, 48(1):56–105, 2019. doi: 10.1080/09298215.2018.1511736.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5045–5054. PMLR, 2018.

Abhinav Verma, Hoang M. Le, Yisong Yue, and Swarat Chaudhuri. Imitation-projected programmatic reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jiakang Wang, Runze Liu, Fuzheng Zhang, Xiu Li, Guorui Zhou, Kun Gai, and Ling Pan. Stabilizing knowledge, promoting reasoning: Dual-token constraints for RLVR. *arXiv preprint arXiv:2507.15778*, 2025a.

Pengcheng Wang, Xinghao Zhu, Yuxin Chen, Chenfeng Xu, Masayoshi Tomizuka, and Chenran Li. Residual policy gradient: A reward view of KL-regularized objective. *arXiv preprint arXiv:2503.11019*, 2025b.

Yusong Wu, Tim Cooijmans, Kyle Kastner, Adam Roberts, Ian Simon, Alexander Scarlatos, Chris Donahue, Cassie Tarakajian, Shayegan Omidshafiei, Aaron Courville, Pablo Samuel Castro, Natasha Jaques, and Cheng-Zhi Anna Huang. Adaptive accompaniment with ReaLchords. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53328–53345. PMLR, 2024.

Yusong Wu, Stephen Brade, Aleksandra Teng Ma, Tia-Jane Fowler, Enning Yang, Berker Banar, Aaron Courville, Natasha Jaques, and Cheng-Zhi Anna Huang. Generative adversarial post-training mitigates reward hacking in live human-AI music interaction. In *International Conference on Learning Representations (ICLR)*, 2026.

Wen-Chi Yang, Giuseppe Marra, Gavin Rens, and Luc De Raedt. Safe reinforcement learning via probabilistic logic shields. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, pp. 5739–5749, 2023. doi: 10.24963/ijcai.2023/637.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned AI. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15763–15773, 2020.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Cost of anchoring a target dimension

Why the anisotropic anchor is beneficial

A scalar anchor has a cost on every dimension where the reward and the reference disagree. The following idealization makes that cost explicit for a target dimension k. Consider a single state, drop the entropy term ($\alpha = 0$), and let the policy of head k be an arbitrary distribution. Let the reward depend on a^k additively, $r(s,a) = r_k(a^k) +$ terms that do not depend on a^k, and let a' be the reward-maximizing value of dimension k. Let the program choose a different value $a^\star$ on this dimension, with a reward gap $\Delta = r_k(a') - r_k(a^\star) > 0$.

Proposition 1 (Cost of anchoring a target dimension)**.** *If π_k^{sym} puts all its mass on $a^\star$, the distribution of head k that maximizes $\mathbb{E}_{a^k\sim\pi_\theta^k}[r_k(a^k)] - \beta_k \,\mathrm{CE}(\pi_k^{\mathrm{sym}}, \pi_\theta^k)$ satisfies*

$$\pi_\theta^k(a^\star) = \min\big(1,\ \beta_k/\Delta\big), \qquad r_k(a') - \mathbb{E}_{\pi_\theta^k}\big[r_k\big] = \min\big(\beta_k,\ \Delta\big). \tag{4}$$

If π_k^{sym} is stochastic, the reward given up on dimension k is nondecreasing in β_k and tends to $r_k(a') - \mathbb{E}_{\pi_k^{\mathrm{sym}}}[r_k]$ as $\beta_k \to \infty$.

The proof is in Appendix A.1. Suppose that holding the non-target dimensions requires an anchor strength of at least β_{hold}. A scalar anchor then gives up at least $\min(\beta_{\mathrm{hold}}, \Delta)$ of reward on dimension k at every state where reward and program disagree, while an anisotropic anchor with $\beta_k = 0$ gives up nothing there and holds the other dimensions equally well. In the idealization the heads are independent, so any $\beta > 0$ would hold a dimension the reward ignores. In a network the shared backbone carries reward-driven updates into the non-target heads, which is why $\beta_{\mathrm{hold}} > 0$ is needed.

A.1 Proof of Proposition 1

Under the stated assumptions the objective separates across heads, so it suffices to consider head k. Write π for π_θ^k, q for π_k^{sym} and β for β_k, and let

$$J(\pi) \;=\; \mathbb{E}_{a\sim\pi}\big[r_k(a)\big] \;+\; \beta \sum_a q(a)\log\pi(a),$$

which is $\mathbb{E}_\pi[r_k] - \beta\,\mathrm{CE}(q,\pi)$.

Deterministic program. Let $q = \delta_{a^\star}$ and $p = \pi(a^\star)$. The anchor term is $\beta\log p$ and does not depend on how the remaining mass $1-p$ is distributed, so at the optimum all of it is placed on a'. Then $J(p) = r_k(a') - \Delta p + \beta\log p$, which is strictly concave on $(0,1]$ with $J'(p) = -\Delta + \beta/p$. If $\beta < \Delta$ the maximizer is the stationary point $p = \beta/\Delta$; if $\beta \geq \Delta$, $J' \geq 0$ on $(0,1]$ and the maximizer is $p = 1$. The reward given up is $r_k(a') - \mathbb{E}_\pi[r_k] = \Delta p = \min(\beta, \Delta)$. □

Stochastic program. Let $\beta_1 < \beta_2$ with maximizers π_1, π_2, and write $R_i = \mathbb{E}_{\pi_i}[r_k]$ and $S_i = \sum_a q(a)\log\pi_i(a)$. Optimality of π_1 at β_1 and of π_2 at β_2 gives $R_1 + \beta_1 S_1 \geq R_2 + \beta_1 S_2$ and $R_2 + \beta_2 S_2 \geq R_1 + \beta_2 S_1$. Adding the two yields $(\beta_2 - \beta_1)(S_2 - S_1) \geq 0$, hence $S_2 \geq S_1$, and the first inequality then gives $R_1 - R_2 \geq \beta_1(S_2 - S_1) \geq 0$. The attained reward is therefore nonincreasing in β, and the reward given up nondecreasing. As $\beta \to \infty$ the maximizer tends to q, the unique maximizer of $\sum_a q(a)\log\pi(a)$, so the reward given up tends to $r_k(a') - \mathbb{E}_q[r_k]$. □

B ALGORITHM

With several agents, all of them collect one shared rollout, acting in the order described in Section 3.1, and each then updates its own policy from it, running Algorithm 1 with its own program and anchor. Updating all agents after every rollout, rather than one agent at a time, was chosen after an earlier comparison in which it needed roughly ten times fewer samples for the same final performance.

Algorithm 1 APA, one update for one agent

Require: policy π_θ (warm or cold), program π^{sym}, strengths $\beta_1, \ldots, \beta_K$, entropy weight α

$\mathcal{B} \leftarrow \emptyset$
for each step t of the rollout **do**
  observe s_t; sample $a_t^k \sim \pi_\theta^k(\cdot \mid s_t)$ for every head k
  execute π^{sym} on s_t and record its sampled action $\tilde{a}_t$ ▷ label at the visited state
  apply a_t (not $\tilde{a}_t$) to the environment; receive r_t
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, \tilde{a}_t, r_t)\}$
end for
standardize the returns in $\mathcal{B}$ and use them as advantages ▷ no value function
for each epoch and each minibatch of $\mathcal{B}$ **do**
  take a gradient step on Equation 2
end for

C EXPERIMENTAL SETUP IN DETAIL

C.1 TESTBED

Symbolic programs. The programs are short sets of hand-written, stochastic rules; the source code is part of the supplementary material. They read the current chord, the position in the bar and their own previous output. The players are coordinated only through the shared chord: neither program reads the soloist, and the bass does not read the piano. The relations the reward scores are therefore not written into any rule; the policies can learn them because their observations contain the soloist's line and the piano's bottom note (Figure 1).

Piano. The top voice plays a chord tone and prefers small steps from its previous note. Its octave changes only occasionally and independently of the pitch class, and its volume accents the downbeat. The inner voices complete the voicing: together with the top note they state every chord tone, and each voice defined by its pitch class is placed at the nearest pitch matching the pitch class below the one above it, while preferring close intervals such as thirds and fourths. Because these voices depend on each other, each is a separate module that observes the voices above it, so that the learned policy can represent the same joint distribution.

Bass. The bass plays a walking line with one note per beat: chord tones on the strong beats and, toward the end of each bar, a chromatic approach to the root of the next chord. A few further rules keep the line in register and avoid large leaps and repeated notes.

Rendering. For the audio examples, the piano is rendered as a sparser comping pattern and a non-reactive drum part is added. Both affect only the audio and are not part of any rollout.

Head shapes. Table 4 lists the learned modules. The piano is split into a top-voice module and three inner-voice modules so that the voicing is factored: if the voicing were a single categorical choice from a dictionary, an anisotropic anchor would have nothing to be anisotropic across and would coincide with a scalar one. Two further piano modules derive the realized voicing and its bottom note from the voice modules; they have no action space and are never learned.

C.2 THE CORPUS SPLIT

The clone used in the shift experiment is fitted on the subset of the corpus whose progressions use only dominant- and minor-seventh chords (resulting in 69 of 405 tunes and 5,099 bars) restricted

Table 4: Learned modules and their action heads. *Support* is the number of categories in each head's action space; *Reward reads it* marks the heads the joint objective reads; β_k gives the coefficient assigned under the anisotropic conditions (conditions 5, 6 and 8).

Player	Module	Head	Support	Reward reads it	aniso. β_k
Piano	top voice	pitch class (TopPC)	12	yes	β
		octave (TopOct)	4	yes	0
		volume (Vol)	4	no	β
	inner voice 1	pitch class (V1)	12	no†	β
	inner voice 2	pitch class (V2)	12	no†	β
	inner voice 3	pitch class (V3)	12	no†	β
Bass	bass	pitch class (BassPC)	12	yes	β
		octave (BassOct)	3	yes	0

†Not read directly, but the bass term reads the piano's bottom note, which the inner voices determine; see Appendix C.3.

further to the six most frequent chord roots (finally resulting in 39 of 405 tunes and 3,182 bars). Fine-tuning runs on the complement.

We split on chord quality rather than on a random subset of solos deliberately. The held-out channels are functions of the chord, not of the soloist, so a clone fitted on a random subset of solos has still seen essentially the whole harmonic mapping and has no blind spot to expose. Splitting on chord quality gives the clone a vocabulary it genuinely never saw, and holding out roots as well sharpens it further, since the symbolic program computes the legal pitch-class set from the root and is therefore root-equivariant by construction while a cloned network is not.

Because the complement still contains many familiar chords, the measured effect is a lower bound: the metrics average over bars the clone handles correctly together with bars it does not.

C.3 Objectives

Let $\mathrm{tent}(x;\, t, s) = \max\big(0,\, 1 - |x - t|/s\big)$, which is 1 at the target t and falls linearly to 0 at distance s. With pitches in MIDI semitones, let m_{solo} be the soloist's lowest sounding note (carried across rests), m_{top} the piano's top voice, m_{bot} the piano's bottom voice and m_{bass} the bass note. Then

$$r_{\mathrm{piano}} = \mathrm{tent}\big(m_{\mathrm{solo}} - m_{\mathrm{top}};\ 2,\ 18\big), \qquad r_{\mathrm{bass}} = \mathrm{tent}\big(m_{\mathrm{bot}} - m_{\mathrm{bass}};\ 12,\ 18\big),$$
$$r_{\mathrm{joint}} = \tfrac{1}{2}\, r_{\mathrm{piano}} + \tfrac{1}{2}\, r_{\mathrm{bass}}, \qquad r_{\mathrm{patched}} = \tfrac{1}{2}\, r_{\mathrm{joint}} + \tfrac{1}{2}\, \mathbb{1}\big[\text{top note is a chord tone}\big].$$

All conditions optimize r_{joint}; the reward-patch conditions optimize r_{patched}.

Why tents. The inequalities one would write first leave no headroom. "Bass below the piano's bottom voice" is already satisfied 87% of the time by the bass program, and a clearance of at least seven semitones 77% of the time, so anchored and unanchored conditions would converge for a trivial reason. A one-sided tracking term fails the other way: the piano program sits above the soloist on most beats, where such a term is flat and supplies no gradient. The two-sided tents score 0.303 (piano) and 0.419 (bass) under the symbolic programs, with a gradient at every state.

C.4 Experiment conditions

Anchors. An anchor to the symbolic program is a cross-entropy between each head's policy and the program's action, sampled and recorded during the same rollout; up to the program's entropy this is forward KL, and the reverse direction is unavailable because it requires the program's density at the policy's own actions. An anchor to the frozen clone evaluates the clone's distribution at the visited state and is run in both directions. Scalar conditions use one coefficient for all eight heads; anisotropic conditions use the assignment in Table 4.

Condition 2. The unanchored condition with reduced step size runs six cells of learning rate and clipping range, all at or below the selected learning rate:

$$(\text{lr},\, \epsilon) \in \big\{(3{\times}10^{-4},\, 0.1),\ (3{\times}10^{-4},\, 0.05),\ (3{\times}10^{-4},\, 0.02),\\ (10^{-4},\, 0.2),\ (10^{-4},\, 0.1),\ (3{\times}10^{-5},\, 0.2)\big\}.$$

Each cell is plotted as its own point; the condition is summarized by the best competence any cell reaches.

C.5 Measures

Table 5 presents the held-out channels used for the competence assessment.

Table 5: Held-out channels. None appears in any selection rule, and none in any objective except top-note legality in the patched reward (Section 5.4). The first four form the *competence* index.

Channel	Definition	Heads it reads	In index
voicing completeness	the voicing states every chord tone	TopPC, V1–V3	yes
voicing legality	the inner voices are chord tones	V1–V3	yes
top-note legality	the top voice is a chord tone	TopPC	yes
bass legality	the bass note is a chord tone	BassPC	yes
voicing spacing	adjacent intervals of 3–5 semitones	TopPC, V1–V3	no
volume meter	accent if and only if downbeat	Vol	no

C.6 Training protocol and hyperparameter selection

Protocol. All conditions use PPO without a learned value function, since rewards are given at every step (Ahmadian et al., 2024); each step's return is standardized over the rollout and used as its advantage. Every condition runs 400 updates at equal budget, with β swept over seven values from 0.001 to 1 and five seeds per setting. Table 6 lists all settings. The ranking of the frontiers is the same for the other promising hyperparameter settings, which we tested on a single seed.

Table 6: Settings shared by all conditions. The first three were selected on condition 1; the rest were fixed in advance.

Setting	Value
learning rate	3×10^{-4}
PPO epochs per update	5
environment cycles per update	2000
updates	400
clipping range ϵ	0.2
discount	0.9
minibatch size	256
entropy weight α	0
anchor strength β	$\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0\}$
seeds per setting	5
readout	mean over updates 281–300
MLP backbone (shared by heads)	2 dense layers, hidden dims [256, 256]
activation function	ReLU

Hyperparameter selection. Learning rate, epochs per update and environment cycles per update were selected on a $3 \times 3 \times 3$ grid at constant total environment interaction, using the unanchored condition only, so that the selection cannot favor any anchored condition. The rule was fixed before the grid ran: among cells whose tail clipping fraction stays below 0.3, take the highest final joint reward, with ties going to the cheaper cell. Reward alone selects a collapsed policy: the highest-reward admissible cell (`lr3e-4_ep3_rc1000`, reward 0.859) retains zero entropy on its worst head. A baseline that wins by becoming deterministic would make every contrast against condition 1

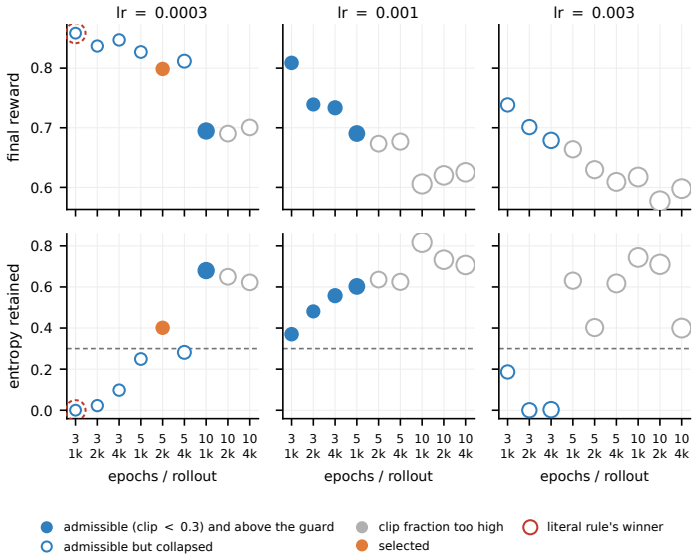


Figure 5: Hyperparameter grid, unanchored condition only. Final reward (top) and retained entropy (bottom) for every cell, one panel per learning rate; marker area shows the tail clipping fraction. Filled markers pass both the clipping criterion and the entropy guard; the dashed ring marks the cell selected without the guard.

look larger than it is. We therefore added one guard, carried over from earlier pilots but not part of the rule fixed in advance: every head must retain at least 0.3 of its clone's entropy. Two cells passing the guard lie within one standard deviation of each other (0.809 ± 0.020 and 0.799 ± 0.010); the tie clause selects the cheaper one, `lr3e-4_ep5_rc2000` (400 updates of 5 epochs against 800 of 3), which also retains more entropy (0.401 against 0.370). Figure 5 shows the full grid.

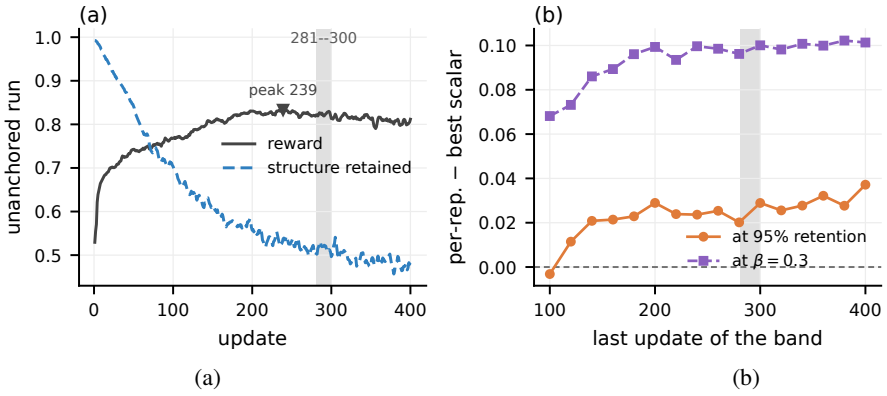
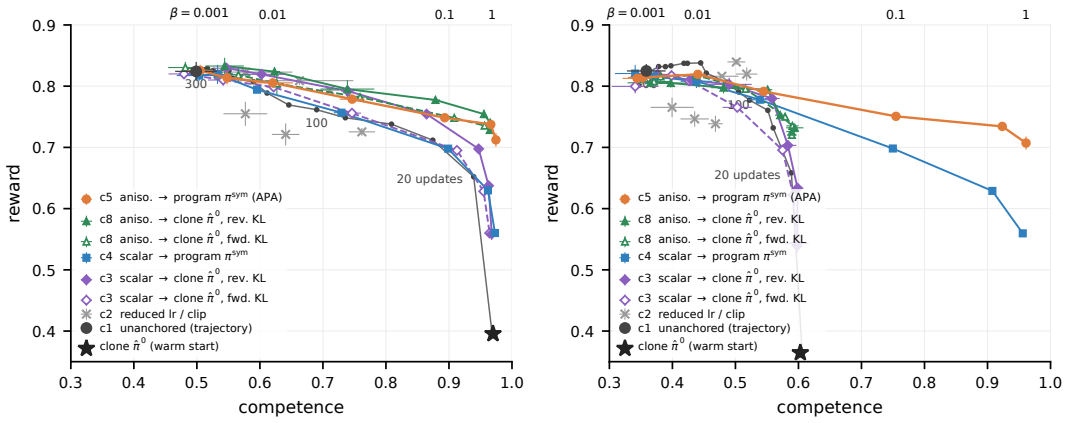


Figure 6: Readout band. **(a)** The unanchored condition at the selected cell: reward peaks near update 250 while competence is still falling. **(b)** Anisotropic minus best scalar condition, recomputed with the end of the band moved across training, at matched competence and at matched β. The shaded strip is the reported band.

Readout band. Runs go to 400 updates and are read at updates 281–300. The unanchored reward peaks near update 250 and declines slightly afterwards, while competence is still falling (Figure 6(a)). Reading on the reward plateau compares against the baseline near its best, which is the conservative choice for every statement about a share of the unanchored reward gain. Figure 6(b) recomputes the main contrast with the end of the band moved from update 100 to 400, at matched competence and at matched β; the difference keeps its sign and approximate magnitude throughout.

D ADDITIONAL RESULTS

D.1 THE FULL FRONTIER IN- AND OUT-OF-DISTRIBUTION



(a) in distribution (b) under shift

Figure 7: Frontier of all warm conditions and all anchor strengths.

Figure 7 draws every run in both settings, and therefore is the complete version of Figure 2a. In both settings, the anisotropic conditions dominate the scalar ones with the same reference, and condition 2 does not reach the high-competence regime. The two settings differ mainly in the ceiling of the clone-anchored conditions discussed in Section 5.2, which is why the main text compares only the anisotropic conditions across settings rather than repeating the full frontier. That ceiling also decides how the shifted frontiers can be compared at all: above a competence of 0.60 the clone-anchored curve does not exist, so there is no matched-competence reading to be had there, and Section 5.2 compares the two at matched β and at matched reward instead.

Direction of the clone anchor. Both directions are run for every clone-anchored condition. Figure 7a puts reverse KL, as the mainstream convention for RLHF-style anchoring, ahead of forward KL on both axes at $\beta = 0.1$ and $\beta = 0.3$ in the scalar and anisotropic families alike, and level with it at $\beta = 1$, which is why the main text compares against that configuration throughout rather than against the weaker one.

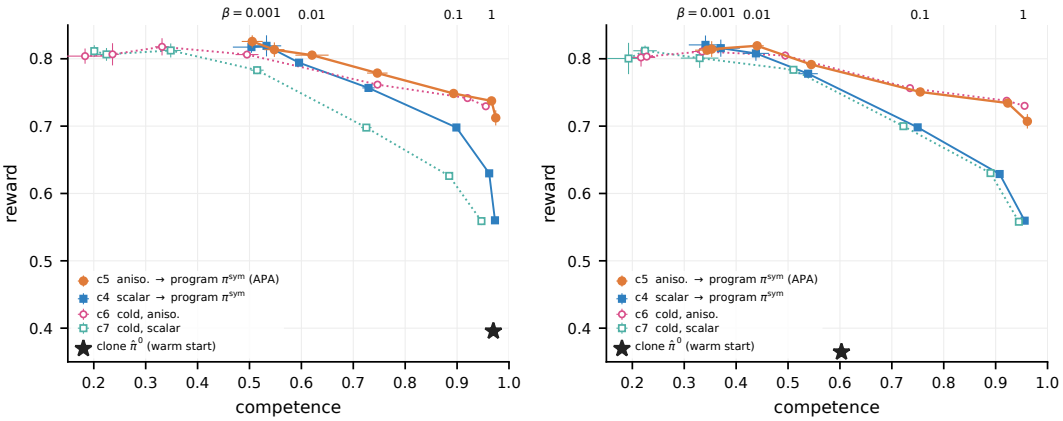
D.2 PER-CHANNEL COMPARISON

Table 7: Joint reward at 95% of the clone's value on each held-out channel, in distribution. The last two rows are channels the competence index does not average.

channel	aniso. program	aniso. clone fwd	aniso. clone rev	scalar program	scalar clone fwd	scalar clone rev
voicing completeness	0.740	0.739	**0.762**	0.647	0.651	0.698
voicing legality	0.746	0.746	**0.770**	0.691	0.696	0.717
top-note legality	0.746	0.747	**0.778**	0.699	0.701	0.735
bass legality	0.753	0.761	**0.780**	0.706	0.702	0.728
volume meter	0.751	0.752	**0.788**	0.705	0.698	0.768
voicing spacing	0.750	0.749	**0.806**	0.713	0.720	0.766

The anisotropic family beats every scalar condition on the four channels of the competence index (Table 7); on the two channels outside it, scalar reverse KL overtakes the anisotropic conditions with forward divergence, while anisotropic reverse KL stays ahead. Within the anisotropic family, and in distribution only, anchoring to the frozen clone with reverse KL is the strongest condition on every channel, by 0.02 to 0.06. We report this rather than averaging it away. In distribution the executable prior buys nothing over a well-fitted clone, and the case for it rests entirely on Section 5.2.

D.3 COLD START



(a) in distribution (b) under shift

Figure 8: Conditions 4–7, all anchor strengths, in both settings.

Conditions 6 and 7 vary the initialization, not the anchor. Two things follow. The anisotropy gap reappears in full from a cold start: at $\beta = 0.3$ in distribution, the cold anisotropic condition returns a reward of 0.742 against 0.626 for the cold scalar one, at higher competence (0.920 against 0.885), so the effect is not an artifact of starting from a cloned policy. And the cold anisotropic condition is competitive with the warm one throughout: under the shift at $\beta = 0.3$ it matches it within the error bars (0.921 / 0.738 against 0.923 / 0.734, competence / reward), and at $\beta = 1.0$ in distribution it returns slightly more reward at slightly lower competence (0.955 / 0.730 against 0.975 / 0.712), while needing a larger β to reach a given competence. That suggests a variant with no cloning phase at all, leaving the executable prior as the only reference in the pipeline; we did not pursue it here. There is deliberately no cold counterpart to the frozen-clone conditions: with no warm start there is no clone to freeze.

D.4 PER-HEAD DRIFT, ALL HEADS, BOTH SETTINGS

Table 8: Per-head drift from π^{sym} at $\beta = 0.3$, every head, in distribution and under the shift. The compact version is Table 3.

setting		condition	TopPC	TopOct	Vol	V1	V2	V3	BassPC	BassOct
in dist.	c1	unanchored	5.37	1.53	0.44	3.19	2.21	2.87	5.86	1.39
	$c3_{\mathrm{rev}}$	scalar → clone rev.	1.31	0.59	0.00	0.74	0.41	0.20	0.78	0.34
	$c3_{\mathrm{fwd}}$	scalar → clone fwd.	1.33	0.56	0.01	0.74	0.41	0.22	0.77	0.35
	c4	scalar → program	1.27	0.55	0.01	0.67	0.34	0.06	0.59	0.31
	$c8_{\mathrm{rev}}$	aniso. → clone rev.	1.26	1.30	0.00	0.72	0.39	0.18	0.80	2.38
	$c8_{\mathrm{fwd}}$	aniso. → clone fwd.	1.27	1.26	0.00	0.72	0.39	0.20	0.79	2.35
	c5	**aniso. → program (APA)**	1.22	1.24	0.00	0.67	0.33	0.06	0.59	2.36
out of dist.	c1	unanchored	6.63	1.63	0.29	4.01	3.20	4.10	6.61	1.29
	$c3_{\mathrm{rev}}$	scalar → clone rev.	2.73	0.58	0.00	2.40	2.46	4.84	2.58	0.56
	$c3_{\mathrm{fwd}}$	scalar → clone fwd.	2.72	0.56	0.00	2.36	2.42	4.39	2.54	0.55
	c4	scalar → program	1.35	0.56	0.01	0.72	0.39	0.13	0.65	0.32
	$c8_{\mathrm{rev}}$	aniso. → clone rev.	2.71	1.33	0.00	2.38	2.49	4.99	2.61	2.60
	$c8_{\mathrm{fwd}}$	aniso. → clone fwd.	2.67	1.40	0.01	2.34	2.40	4.60	2.54	2.58
	c5	**aniso. → program (APA)**	1.28	1.23	0.01	0.69	0.36	0.11	0.63	2.42

The pattern in Figure 3 holds head by head. Under shift every clone-anchored condition moves together — the four agree to within 14% on each held head — while the two program-anchored conditions sit at their in-distribution values. The inner voices separate most: the third inner voice goes from 0.18–0.22 to 4.4–5.0 for the clone-anchored conditions and from 0.06 to 0.11–0.13 for the program-anchored ones.

D.5 VOCABULARY: WHAT RANGE EACH CONDITION KEPT

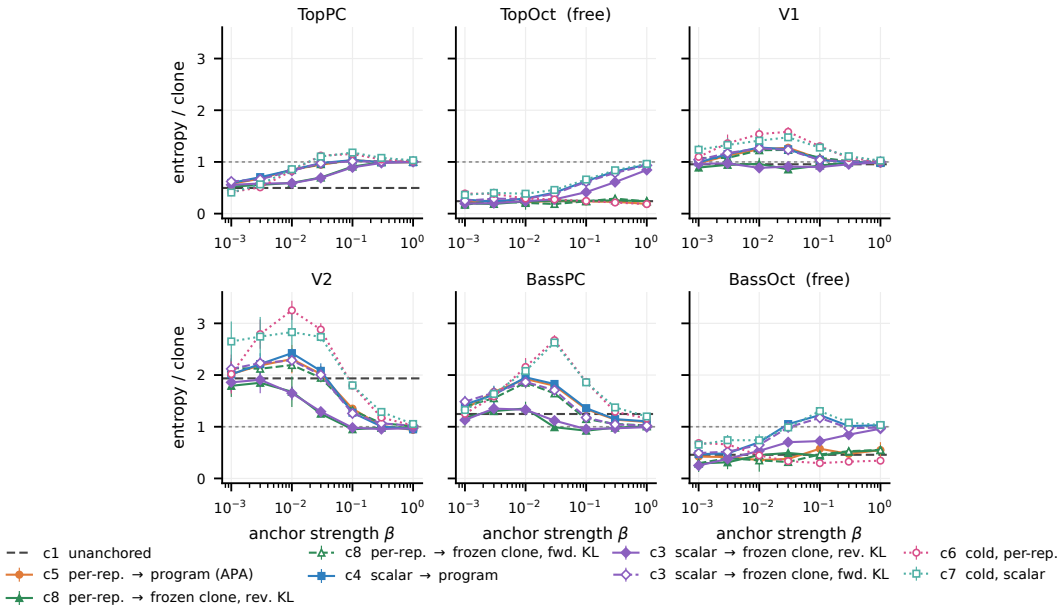


Figure 9: Policy entropy per head, as a fraction of the clone's, against anchor strength.

Distance says how far a policy moved, not whether it still makes choices. We report per-head policy entropy as a fraction of the clone's at the first logged update, computed from the policy's own logits and so independent of the entropy floor: 1.0 is the starting range, below it the policy narrowed and above it diffused. Vol and V3 are omitted as near-deterministic under the program (0.001 and 0.068 nats).

Table 9: Policy entropy as a fraction of the clone's, at $\beta = 0.3$, updates 281–300, five seeds. TopOct and BassOct are the heads the anisotropic assignment leaves at $\beta_k = 0$; a narrower range there is intended rather than damage.

	held by every anchor				free under aniso.	
condition	TopPC	V1	V2	BassPC	TopOct	BassOct
c1 unanchored	0.50	0.95	1.94	1.25	0.24	0.46
c3 scalar → clone (fwd.)	1.00	1.01	1.07	1.04	0.79	0.97
c3 scalar → clone (rev.)	0.98	0.96	0.96	0.97	0.61	0.85
c4 scalar → program	1.00	0.99	1.01	1.15	0.82	1.01
c8 aniso. → clone (fwd.)	1.01	1.02	1.07	1.05	0.29	0.53
c8 aniso. → clone (rev.)	0.99	0.99	0.98	0.98	0.26	0.52
c5 aniso. → program (APA)	0.99	0.99	1.02	1.14	0.23	0.47
c6 cold, aniso.	1.05	1.08	1.18	1.29	0.21	0.32
c7 cold, scalar	1.08	1.11	1.29	1.37	0.84	1.08

The anisotropy is visible without any floor. All eight anchored conditions are indistinguishable on the heads every anchor holds (0.98–1.08 on the top pitch class) and separate without overlap on the register heads, where the anisotropic conditions keep 0.21–0.29 of the clone's range against the scalar conditions' 0.61–0.84. The scalar conditions are still choosing among registers where the reward wants one, which is what they spend their reward on.

The unanchored policy does not simply collapse. It narrows where the reward pulls (0.50 on the top pitch class, 0.24 and 0.46 on the registers) and widens past the clone elsewhere (1.94 on V2, 1.25 on BassPC), so a penalty toward a point estimate would catch only half the damage; APA holds both sides at once (0.99–1.14).

Reverse KL holds a narrower vocabulary. Being mode-seeking, it keeps competence by concentrating on part of the reference rather than covering it, and sits below its forward-KL counterpart at every anchor strength in both families: by 0.38 (scalar) and 0.29 (anisotropic) at $\beta = 0.03$, falling to 0.09 and 0.04 at the $\beta = 0.3$ operating point — too small there to explain the frozen clone's slight in-distribution advantage in Table 7.