WHEN CAN A WORLD MODEL DECIDE? CERTIFYING ELITE DECISIONS IN SAMPLING-BASED PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sampling-based planners such as CEM and MPPI turn a latent world model into a controller by trusting its ranking of candidate action sequences at every iteration. We show when that trust is warranted. A rollout error bound yields a decision certificate: the model places a candidate correctly in or out of the elite set whenever the candidate lies more than twice the error from the elite boundary. Executing every candidate a planner evaluates on Push-T, we find that the certified fraction falls from 0.67 to 0.29 over the search: the planner converges until its candidates differ by less than the model can resolve, and more search cannot recover the lost resolution. The certificate also names the remedy, a more accurate predictor or models with independent errors, and only the decisions at the elite boundary need it. We introduce EliteCert, in which the planning model screens every candidate and independently trained validator models re-score only a calibrated band around the K-th elite. EliteCert enters through the cost function, so CEM, iCEM and MPPI run unmodified. Under the official LeWM protocol it raises success by 9–14 points for all three planners on Push-T and by 2–3 points on Cube, beats the default configuration at 44% of its CPU time, matches re-scoring every candidate at a third of the extra compute, and adds a further 1.8 points to a planner whose predictor we retrained. An offline qualification test tells in advance which models can serve as validators. Finally, we prove and measure that gains on logged candidate pools do not transfer to the closed loop of an iterated planner, and evaluate every claim in the closed loop, with 1800 paired runs behind each headline comparison.

1 INTRODUCTION

World models trained from pixels are increasingly used for control by planning in their latent space (Hafner et al., 2019; Hansen et al., 2022; Zhou et al., 2025). The planner is usually a population-based optimizer: CEM (Rubinstein & Kroese, 2004) or MPPI (Williams et al., 2017) samples candidate action sequences, rolls each through the world model, keeps the lowest-cost elites, and refits its sampling distribution. JEPA-style world models make each rollout cheap (Maes et al., 2026), and planners are run with generous budgets; LeWM, for example, plans Push-T with 300 candidates refined over 30 iterations.

Every iteration of such a planner trusts the model's ranking, and that trust erodes exactly when the planner succeeds. As the search converges, its candidates become similar, their cost differences shrink toward the model's own error, and the ranking that drives the update becomes noise. The standard responses do not address this. Larger budgets keep refining candidates the model can no longer tell apart, and ensembles in model-based RL (Chua et al., 2018) propagate every candidate through every member, multiplying the cost of the one component, the rollout, that planners are designed to spend freely. What is missing is a way to tell which decisions a model can make, and to spend extra models only on the ones it cannot.

We provide that tool (Section 3). A standard rollout-error chain bounds every candidate's value error by some ε that grows with the predictor's one-step defect, and it follows that the model places a candidate correctly in or out of the elite set whenever the candidate lies more than 2ε from the elite boundary. Executing every candidate a planner evaluates in the simulator (Section 4), we measure the

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.

certificate directly during planning: on Push-T the fraction of candidates whose elite membership is certified falls from 0.67 at the first CEM iteration to 0.29 at the last, and plan quality saturates halfway through the default budget. The certificate also names two levers: a predictor with a lower one-step defect, and averaging models whose errors are partly independent, which lowers the error of value *differences*.

Our method, EliteCert (Section 5), pulls the second lever where it matters. A population planner uses its costs only to split candidates into elites and the rest, so only decisions at the elite boundary can change the plan. EliteCert lets the planning model screen every candidate and has independently trained validators re-score the resolution band, the candidates whose standardized cost lies within δ of the K-th. The band width is read off a coverage target on measured promotion gaps, an offline qualification test decides which models may act as validators, and the whole procedure enters through the cost function, so CEM, MPPI and iCEM run unmodified. It works because it spends the second opinion on the decisions the planner cannot certify and on nothing else: MC-dropout copies of the planner, whose ranking errors stay correlated with its own, give no gain, while independently trained models do.

Under the official LeWM protocol, EliteCert adds 8.7, 12.4 and 13.8 points to CEM, iCEM and MPPI on Push-T and 2–3 points on Cube (Table 1), beats the default configuration at 44% of its CPU time, and matches re-scoring every candidate at a third of the extra compute. The two levers compose: retraining only the predictor on the planner's frozen encoder raises success from 82.5% to 89.3%, and certifying that planner adds a further 1.8 points.

The certificate also explains a trap that recurs in this line of work (Section 7): a population planner generates the candidates it compares, so an improvement measured on pools logged from the unmodified planner says nothing about the pools the modified planner will create. We prove this in both directions and document five measured instances, one with the offline and closed-loop orderings reversed.

Contributions.

- A decision certificate for population-based planners with world models (Theorem 1) and a ground-truth measurement of how many elite decisions a model certifies during planning, showing that resolution, not search, limits late iterations.
- EliteCert, a planner-agnostic, training-free procedure that spends independent models only on uncertified elite decisions, with a band calibrated to a coverage target, an offline qualification test for validators, and an exact reduction to the native planner when none qualifies. It improves CEM, iCEM and MPPI by 9–14 points on Push-T and composes with a retrained predictor.
- A proof and five measured instances that offline and closed-loop improvement are independent for iterated planners, and a closed-loop evaluation paired by task over three environments, three planners and three planner seeds.

2 Background: population-based planning in latent space

A latent world model encodes an observation o_t and a goal image g into embeddings $z_t = f_\theta(o_t)$, $z_g = f_\theta(g)$, and predicts future embeddings autoregressively, $\hat{z}_{t+h+1} = p_\theta(\hat{z}_{t+h-2:t+h}, a_{t+h-2:t+h})$. The cost of an action sequence $a = (a_1, \dots, a_H)$ is the terminal distance $C_\theta(a) = \|\hat{z}_{t+H} - z_g\|_2^2$. CEM samples N candidates $a^{(i)} \sim \mathcal{N}(\mu, \operatorname{diag}\sigma^2)$, evaluates C_θ, keeps the K lowest-cost elites E, and sets μ, σ to their mean and standard deviation; after T iterations it executes μ. MPPI weights the elites by $\operatorname{softmax}(-C/\lambda)$ and keeps σ fixed. In both, *only the identities of the elites and their costs* enter the update. We use the planners and pretrained LeWM models of the `stable-worldmodel` library unchanged (Maes et al., 2026); our re-implementation reproduces the native planners' actions bit-for-bit when EliteCert is disabled.

3 WHEN CAN A WORLD MODEL DECIDE?

A planner acts on the model's ranking of candidates, so the question is when that ranking can be trusted. Let $Q(a)$ be the true value of a candidate action sequence and $\hat{Q}(a)$ the value the world model assigns it through an H-step rollout. The error of $\hat{Q}$ is controlled by the model's one-step error through a standard chain (proof in Appendix A).

Theorem 1 (Rollout and decision certificate)**.** *Let the latent predictor T be L-Lipschitz with one-step defect κ against the encoded true dynamics, and let the cost read-out be ℓ-Lipschitz with read-out error η. Then every candidate's value error satisfies $|\hat{Q}(a) - Q(a)| \le \varepsilon_H := \eta + \ell\kappa \sum_{j<H} L^j$. Consequently the model ranks two candidates correctly whenever their true values differ by more than $2\varepsilon_H$, and it places a candidate correctly inside (outside) its top K whenever the candidate's true value exceeds the $(K{+}1)$-th best by more than $2\varepsilon_H$ (falls short of the K-th best by more than $2\varepsilon_H$). The same holds with true and model values exchanged (model-side form).*

The theorem says what a model can and cannot decide, and it names two ways to make it decide more. A decision is *certified* when the gap it rests on exceeds 2ε. The model-side form needs only the model's costs and ε, and is the one we count. The bound can be made smaller by lowering the one-step defect κ, which is a property of the predictor, or by lowering the error of the *difference* between two candidates' values, which is what averaging models with partly independent errors does: with $M{+}1$ models whose pairwise difference errors have equal variance and correlation ρ, the variance falls by $(1 + M\rho)/(M + 1)$. Section 4 measures how many elite decisions are certified during a planning run, and the rest of the paper pulls on both levers.

A certificate concerns the candidates the planner actually compares, and a population planner generates those itself, so an improvement on pools logged from the unmodified planner certifies nothing about the pools a modified planner will generate.

Proposition 1 (Offline and closed-loop improvement are independent)**.** *For an iterated population planner, a scorer that selects strictly better elites than the planning model on every pool logged from the native planner need not improve the closed-loop outcome, and a scorer that selects strictly worse elites on some logged pool can improve it.*

Appendix A gives a two-iteration witness for each direction. Section 7 reports five measured instances of the first direction.

4 HOW MANY DECISIONS DOES THE MODEL CERTIFY?

Ground-truth audit. We log every candidate evaluated by native CEM ($N{=}300$, $T{=}30$, $K{=}30$) during closed-loop Push-T control and execute each candidate from the true start state in the simulator: a first audit of 70 decisions from 50 tasks, later extended to 561 decisions from 400 episodes. The outcome is scored by the environment's own success test through a margin $m = \max(\|\Delta p\|/20,\ |\Delta\theta|/20^\circ)$, which is below 1 exactly when the task counts as solved.

Figure 2 and Table 3 (Appendix B) show that the model's ranking agrees less and less with reality as the search proceeds: its rank correlation with the executed outcome falls from 0.74 at the first iteration to 0.23 at the last, because late candidates are close to each other and their cost differences fall below the world model's error. The real quality of the plan stops improving after about 15 of the 30 iterations, while the elites chosen with three independent models keep improving (Figure 2, right). Closed-loop results agree: over 200 tasks, $300{\times}15$ reaches the same success as the default $300{\times}30$ (77.5% on CPU), and $150{\times}15$ is 1.5 points below.

Counting certified decisions. Theorem 1 turns this into a count (Figure 1). On the full 561-decision audit we fit each pool's standardized model cost as an affine function of the standardized true outcome, take the median absolute residual in each phase of the search as the empirical ε, and call a candidate's elite membership certified when it lies more than 2ε from the model's K-th cost; this is an empirical analogue of the theorem, not a bound, and leaves 4–12% of certified decisions wrong. The planner certifies 0.67 of the candidates at the first iteration and 0.29 at the last, and the true top 30 are certified as elites in under 1% of cases after iteration 5. Independent models raise the

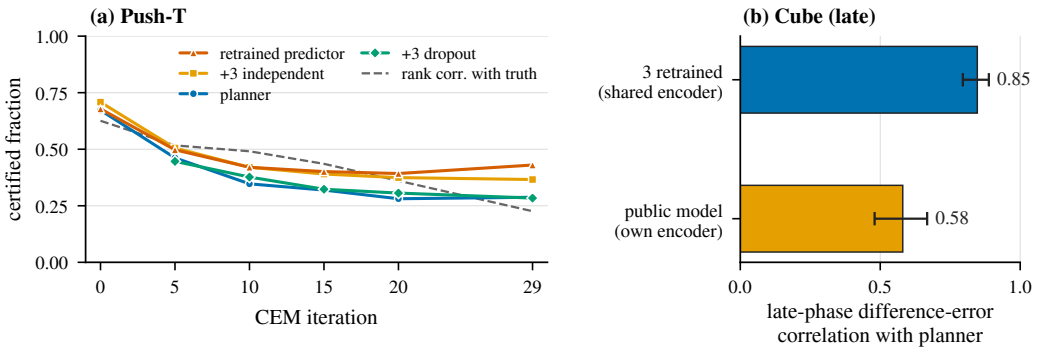


Figure 1: (a) Push-T, 561 decisions with every candidate executed: fraction of candidates whose elite membership is certified (lie more than 2ε from the K-th cost, ε the median residual against the true outcome), for the planner, the planner averaged with three independently trained models or with three MC-dropout copies (70 decisions), and one predictor retrained on the planner's frozen encoder; dashed: the planner's rank correlation with the true outcome. (b) Cube, 51 decisions: late-iteration correlation between the planner's difference errors and those of three predictors retrained on its encoder or of a model with its own encoder (95% bootstrap intervals).

late fraction to 0.37 and halve the error rate among certified decisions ($0.12 \to 0.07$), a retrained predictor raises it to 0.43, and MC-dropout copies leave it at 0.28. Appendix B gives all rows.

More search does not fix it. Truncated rollouts, smoothing costs over neighbouring candidates and adaptive stopping all fail to use the surplus iterations (Appendix C).

5 EliteCert: certifying the elite boundary

Only the boundary decides. A population-based planner uses its costs in exactly one way: to split the N candidates into the K elites that refit the sampling distribution and the $N - K$ that are discarded. A ranking error therefore changes the plan exactly when it moves a candidate across the K-th position. EliteCert concentrates all of its extra computation on these decisions, and its design answers one question: which candidates are they?

The resolution band. Let $c_1..c_N$ be the planning model's costs for one iteration and $z_i = (c_i - \bar{c})/s_c$ their standardization inside the pool, with $z_{(K)}$ the K-th smallest. EliteCert certifies

$$B_\delta = \{\, i : z_i \le z_{(K)} + \delta \,\}, \tag{1}$$

re-scores B_δ with M independently trained validator world models $\{C_{\phi_m}\}$, standardizes each model's costs over B_δ, and takes as elites the K members of B_δ with the lowest mean standardized cost (Algorithm 1). Candidates outside B_δ keep the planner's ranking and cannot become elites. This is certification in the sense of Theorem 1: on the band, where the planner's own margin is too small to decide, averaging models with partly independent errors lowers the ε the decision rests on and raises the certified fraction (Figure 1).

The single parameter δ is measured in units of the planner–validator disagreement, and the following statement shows why a band is the right object (proof in Appendix A).

Proposition 2 (Band sufficiency)**.** *Let s be any score on a candidate set containing the planner's top K. If there are constants $a > 0$ and b with $|s_i - az_i - b| \le \varepsilon$ for every candidate, then no candidate with $z_i > z_{(K)} + 2\varepsilon/a$ is among the K smallest values of s. Hence taking the K smallest s inside $B_{2\varepsilon/a}$ returns exactly the elites of ranking every candidate by s.*

The deployed score standardizes each model over the band, which makes the planner's own term exactly affine in z, so the proposition applies to it directly. In practice promotions lie much closer to the boundary than a uniform bound allows (fitted $2\varepsilon/a$: median 0.52; promoted elites: median 0.14), so we calibrate δ on where promotions actually occur. A compute guard caps $|B_\delta|$ at $\max(K, N/2)$ and never binds.

Algorithm 1 One EliteCert iteration (CEM shown; MPPI and iCEM identical up to the update)

1: sample $a^{(1..N)} \sim \mathcal{N}(\mu, \operatorname{diag}\sigma^2)$, set $a^{(1)} = \mu$
2: $c_i \leftarrow C_\theta(a^{(i)})$ for all i {planning model, batched}
3: $z \leftarrow (c - \bar{c})/s_c$; $\quad B \leftarrow \{i : z_i \le z_{(K)} + \delta\}$ {resolution band, Eq. (1)}
4: **for** $m = 1..M$ **do**
5: $\quad c_i^m \leftarrow C_{\phi_m}(a^{(i)})$ for $i \in B$ {validators, $|B|$ rollouts each}
6: **end for**
7: $s_i \leftarrow \frac{1}{M+1}\big(z(c)_i + \sum_m z(c^m)_i\big)$ for $i \in B$ {$z(\cdot)$ standardizes over B}
8: $E \leftarrow$ the K indices in B with the smallest s_i; $\mu, \sigma \leftarrow$ mean, std of $\{a^{(i)}\}_{i \in E}$

Calibrating δ to a coverage target. The band should contain the candidates that certification promotes, and this is a quantity we can calibrate directly. On calibration pools, record for every candidate the certified score lifts into the top K its *promotion gap*, the distance $z_i - z_{(K)}$ by which the planner had placed it outside, and set δ to the $(1-\alpha)$ quantile of those gaps. At the configuration we deploy, nominal coverages of $0.80/0.90/0.95$ give $\delta = 0.30/0.48/0.69$ and measured coverages of $0.78/0.87/0.91$ over bands holding $26/35/43\%$ of the pool; the small shortfall against nominal reflects gaps that cluster within pools (Appendix D). δ is stable in the calibration set: from a quarter of the episodes it is 0.49 ± 0.07 over 200 resamples. On pools logged from a certified run, the distribution the band is deployed on, coverage is higher at every level ($0.82/0.93/0.96$) over slightly narrower bands: certification reduces the disagreement it corrects (promotions per pool fall from 4.7 to 4.2), so calibrating on native pools is conservative. Appendix D gives the full tables.

Why a band. Certifying the planner's top rK fixes how many candidates are verified but not which, and the two come apart because the spread of the cost pool changes by an order of magnitude over a run: at iteration 29 the top $2K$ retains only 25% of the candidates certification would promote, while promotions per pool grow from 5.3 to 22.4. The band adapts to this automatically and matches or beats both a fixed count and a margin gate (85.0% at the best of three swept thresholds against 85.5%) without any sweep.

Why independent models. By Section 3, averaging helps only through the error of value differences, which shrinks by $(1 + M\rho)/(M + 1)$. On the audit the validators' difference errors correlate with the planner's at $0.79/0.70/0.54$ in early/middle/late iterations, while three MC-dropout copies stay at $0.93/0.88/0.71$ on the 70 decisions where they were logged, and averaging them does not reduce the planner's error (variance ratio 0.99–1.01, against 0.82–0.91 for the validators; Appendix A). Dropout, shifted views of one model, and learned heads on the planner's own predicted trajectory therefore give no benefit (Appendices C and E).

Which models may certify? Averaging only helps if a validator discriminates between nearby plans at least as well as the planning model; a weaker validator moves candidates across the boundary for the wrong reasons. We qualify models offline, before deployment and without environment interaction: on held-out dataset windows, each model scores the action sequence that was actually executed against 31 perturbations of it at scale σ (in normalized action units, σ=0.1 matching the late-iteration spread of CEM), and we record the mean percentile rank q_σ of the executed sequence. A model is admitted as validator if $q_{0.1} \le q_{0.1}(\text{planner})$. If no model qualifies, EliteCert reduces to the native planner.

Exactness and cost. With $M = 0$, EliteCert reproduces the library's CEM and MPPI actions bit-for-bit under the same random stream. On GPU three or more validators run in one vectorized call (`torch.func.vmap` over stacked parameters), on CPU one after another (Appendices F and G).

6 Experiments

Setup. On Push-T the planning model is the official LeWM checkpoint and the validators are three publicly released LeWM checkpoints trained with different seeds on the same data; Cube has one further public checkpoint; for Reacher we trained three additional models ourselves. Two task

Table 1: Official LeWM protocol, 150×10, band $\delta=0.481$, planner seeds 42/7/13; 600 tasks per seed on Push-T (three validators) and Cube (one), 200 on Reacher, where the three independently trained checkpoints fail the qualification test, so EliteCert with them reduces to the native planner (ungated: forced in as validators; for predictors retrained on the planner's encoder see Section 6.6). Δ with 95% task-cluster bootstrap interval; p: McNemar over (seed, task) units, descriptive since seeds share tasks.

Env	Planner	Native	EliteCert	Δ [95% CI], p
Push-T	CEM	82.5	91.2	$+8.7$ [$+6.8$, $+10.5$], 2×10^{-26}
Push-T	iCEM (unmodified)	66.1	78.5	$+12.4$ [$+9.9$, $+15.0$], 4×10^{-27}
Push-T	MPPI	63.9	77.7	$+13.8$ [$+11.2$, $+16.3$], 6×10^{-28}
Cube	CEM	66.4	69.4	$+2.9$ [$+1.5$, $+4.5$], 4×10^{-5}
Cube	MPPI	45.8	47.9	$+2.1$ [$+0.2$, $+3.9$], 0.025
Reacher	CEM	65.5	65.5 (gate abstains)	ungated -6.3, 0.007

Table 2: Official LeWM protocol, 600 Push-T tasks, CEM, three validators, one planner seed, paired by task against the default configuration; wall-clock is a separate single-process measurement (40 tasks, 8 threads), since the 600-task runs were sharded across machines.

Configuration	Boundary set	Success	vs. default	median s	rel. time
Native 150×10	—	82.50	-3.2	1.97	0.20
Native 300×30 (default)	—	85.67	—	9.73	1.00
EliteCert, fixed count $r=2$	$rK=30$	90.67	$+5.0$	3.58	0.37
EliteCert, band $\delta=0.296$	80% coverage	91.17	$+5.5$ ($p=6\times10^{-4}$)	3.98	0.41
EliteCert, band $\delta=0.481$	90% coverage	**92.00**	$+6.3$ ($p=4\times10^{-5}$)	4.29	0.44

protocols are used. The official LeWM protocol (tasks sampled uniformly from dataset states with seed 42; 600 tasks on Push-T and Cube, 200 on Reacher; planner seeds 42, 7 and 13) carries the headline comparisons of Section 6.1. The compute frontier, controls and ablations of Sections 6.2–6.5 use, unless stated, a harder in-house Push-T protocol: 200 tasks starting at step 15 of held-out dataset episodes with the goal 25 steps ahead and a 50-step budget, on which native 150×10 CEM reaches 74.5% against 82.5% under the official protocol. Every configuration is evaluated on the same tasks, so arms are compared by paired differences with McNemar tests and task-cluster bootstrap intervals. Intervals resample whole tasks together with all their planner seeds. Compute is reported as world-model rollouts per decision (planner plus validators) and as wall-clock on a CPU node and an RTX 4060 laptop GPU (details in Appendix F).

6.1 Main result under the official protocol

Table 1 is the headline comparison: three planners on Push-T, two on Cube, three planner seeds each, with δ read off a 90% coverage target (Section 5) and not tuned. EliteCert adds 8.7, 12.4 and 13.8 points to CEM, iCEM and MPPI on Push-T with three validators, and 2–3 points on Cube, where a single public validator exists. On Reacher the gate does its job: the three models we trained independently fail the qualification test, and forcing them in as validators would have cost 6.3 points ($p=0.007$, Appendix G); predictors retrained on the planner's encoder pass at equality (Section 6.6). Before running Table 1 we registered a prediction from the planners' update rules (Section 6.4): on Push-T the gains would be ordered CEM $<$ iCEM $<$ MPPI. They are, with iCEM and MPPI gaining 3.7–5.1 points more than CEM; the companion prediction that MPPI would also gain more than CEM on Cube did not hold (-0.8 [-3.2, $+1.5$]).

Against the default configuration. Table 2 compares with the default LeWM configuration, 300×30, on the same 600 Push-T tasks, with wall-clock measured single-process. The band is 3.8 points better over three planner seeds (91.2 against 87.4, $p=10^{-5}$) at 44% of the default's CPU time, and 6.3 points on the seed shown (Appendix G). It also edges out a fixed count tuned for this task ($+1.3$ points at 20% more time) while needing no tuning: its width follows from a coverage target.

6.2 SUCCESS VERSUS COMPUTE

Figure 3 (Appendix H) plots the frontier on the in-house protocol and Table 10 lists the paired comparisons. Across seven CEM budgets (Appendix G), EliteCert adds 4–10 points for CEM, 12–17 points for MPPI, and 16–18 points for iCEM, each at one planner seed. iCEM runs unmodified: EliteCert enters as a cost-function wrapper that returns certified scores on the band and $+\infty$ elsewhere. The best native CEM configuration (300×15, 4,500 rollouts per decision, 77.5%) is exceeded by EliteCert at 150×10 with one validator (1,800 rollouts, 83.0%).

Wall-clock. On CPU time is proportional to rollouts and EliteCert is $4\times$ faster than the default 300×30 at higher success; on a laptop GPU rollouts are latency-bound, so the advantage is clearest on CPU (Table 6).

6.3 CERTIFICATION AND A BETTER MODEL COMPOSE

The validators on Push-T are themselves good planners: as planning models they reach 77.0% and 79.0% (seeds 0 and 1, 150×10) against 74.5% for the official checkpoint. Table 7 in Appendix G therefore promotes each of them to planner in turn. With the strongest of them planning (seed 1, 79.0%), certifying with two further models adds 5.5 points (p=0.052) and with three 6.5 points (p=0.011), reaching 85.5% — above every single model on its own. With a single validator the gain depends on which model it is ($+2$ to $+8.5$ points), so we recommend $M \geq 2$.

Validators from the planner's own checkpoint, and the strongest baseline. Validators need not be separately trained world models. Retraining only the predictor on the planning model's frozen encoder latents (three seeds, a few CPU hours each) gives models that pass the qualification test on Push-T ($q_{0.1}$ 0.26–0.29 against 0.36) and whose difference errors correlate with the planner's at $0.84/0.71/0.54$ (early/middle/late), as the public checkpoints' do ($0.79/0.70/0.54$): the independence on Push-T comes from retraining the predictor, not from a different encoder. As validators under the official protocol they raise success from 82.5 to 91.6% ($+9.1$ $[+7.2, +11.1]$, 1800 units), indistinguishable from the public validators ($+0.4$, p=0.55). They are also better planners than the official predictor (held-out error 0.0045 against 0.0079): planning with one of them alone reaches 89.3% ($+6.8$), the first lever of Theorem 1, and certifying it with the official model and the other two adds a further $+1.8$ $[+0.5, +3.1]$ (p=0.005, 1800 units), the second. The two levers compose, and both are worth pulling. The first is strongest on Push-T, where retraining cuts one-step error by 43%; on Cube the retrained predictors share the planner's encoder and its errors, and the gain comes from a validator with its own encoder (Appendix I).

Does architectural diversity help? Three GRU predictors on the same latents also qualify and add 6.3 points, fewer than the retrained transformers (p=0.04), and mixing the two families does not beat the transformers alone (Appendix I).

6.4 WHY THE GAIN DIFFERS ACROSS PLANNERS

Table 9 (Appendix H) certifies CEM, iCEM and MPPI, all at library defaults, at a matched budget over six planner seeds. The gains are ordered by how much a mis-ranked candidate can move each update: CEM's top-K truncation absorbs errors that stay outside the elite set; iCEM truncates too but carries elites across iterations, so a wrong promotion compounds; MPPI weights elites by $\exp(-C/\lambda)$, so a wrongly promoted candidate can dominate. The measured gains follow that order: $+11.5$, $+13.6$, $+16.4$ points over 1200 (seed, task) units each.

6.5 THE BAND: CALIBRATION, WHAT IT SAVES, AND ABLATIONS

The band has to be calibrated once, and success is flat over an order of magnitude in δ: between 0.1 and 1.0 it moves by 3.3 points on GPU (82.8 to 86.1) and 2.2 on CPU (82.8 to 85.0), with no sharp optimum on either (Appendix D). The band therefore needs a scale, not a search.

What the band buys is visible against the upper bound of certifying every candidate. On the 200-task evaluation (CEM 150×10, three validators) certifying everything and the band at δ=0.1 both

reach 85.5%, differing on twelve tasks, six each way (p=1.00), against 74.5% native. Timed single-process on 40 tasks they cost $3.98\times$ and $2.00\times$ the native planner's wall-clock, so two thirds of full certification goes on candidates the planner had already placed unambiguously.

Ablations. Table 11 (Appendix J) varies the number and identity of validators and the boundary size; every variant beats native, and certifying only the second half of the iterations keeps most of the gain at half the overhead. Two alternative rules for turning the certified scores into the next sampling distribution leave success unchanged (89.7, 89.2, 89.0% over 600 units): what matters is which candidates get certified.

A held-out confirmation. All of the above tunes nothing, but the 200-task set has been looked at repeatedly. We therefore also report a set of 80 Push-T episodes held out from every fit and every choice in this paper, including the two learned scorers of Appendix E, evaluated at 300×30 with three planner seeds and pairing on (seed, episode). Certification raises success from 72.5% to 85.0%: $+12.5$ points, 95% bootstrap interval $[+7.1, +17.9]$, 40 of the 240 units won against 10 lost, McNemar $p = 2.4 \times 10^{-5}$.

6.6 Validator quality and the qualification gate

Table 8 (Appendix H) gives the official-protocol results per environment together with the offline qualification scores. On Push-T a single validator already reaches 89.3%, at a quarter of the default's CPU time (Table 6). On Cube, with the one other public checkpoint as validator, a fixed boundary count of $2K$ gains 2.7 points over 600 tasks (p=0.02; unresolved at 200), and the Push-T band transferred unchanged gains 3.0: the band's δ carries over between environments without recalibration, and every planner seed resolves the gain (Appendix G). On Reacher the three checkpoints we trained ourselves fail qualification: they are weaker planners (52–61% vs. 68.5% for the official model), and forced in as validators they *lower* success (Section 6.1).

The offline qualification test of Section 5 gets all three signs right without a closed-loop run: the Push-T and Cube validators rank the executed action sequence better than the planning model does ($q_{0.1}$ 0.27–0.29 vs. 0.37 on Push-T, 0.17 vs. 0.35 on Cube), the Reacher validators slightly worse (0.35 vs. 0.34). It predicts the sign of the gain; the size also depends on how many validators exist, and Cube, with the widest margin, has one. Predictors retrained on the planner's encoder pass the Reacher test only at equality (0.34 both) and are admitted; they give $+1.2$ points at one seed (p=0.66) and, certifying the last five iterations, $+2.2$ over three (p=0.059; $+5.2$, $+2.7$, -1.2 per seed): small and seed-sensitive (Appendix I).

7 Why every claim here is closed-loop

Proposition 1 describes something we observed repeatedly. Five times in this work a change was clearly better on data or candidate pools logged from the unmodified planner, and none of the five produced a resolved closed-loop gain on the same tasks (Appendix E). (i) A scorer supervised on real executed outcomes raised the true quality of the selected elites from 0.567 to 0.625 offline and lost 22.5 points in the loop. (ii) Dropping the planning model's own cost from the certified score raised offline elite quality by 3.7 points and changed success by $+1.0$ and -1.2 points at two band widths, neither significant. (iii) A residual memory of observed innovations raised true elite quality at every CEM iteration and changed success by -0.2 points. (iv) A recurrent control with the same parameter budget raised offline elite quality 2.5 times as much as the memory and was 2.5 points worse in the loop, so between (iii) and (iv) the offline and closed-loop orderings are reversed. (v) Refitting the memory's geometry and aligning its training to deployment lowered five-step latent error by 14% and lost 4.0 points (p=0.03). The certificate says why: it holds for the candidates the planner compares, and a change to the scorer or the model changes which candidates those are. Our offline quantities (coverage, error correlations, certified fractions) describe the screen and the models; every success claim here is closed-loop.

8 Related work

Planning with learned world models. PlaNet and Dreamer plan or learn behaviours in learned latent spaces (Hafner et al., 2019; 2020); TD-MPC and TD-MPC2 combine MPPI with learned values (Hansen et al., 2022; 2024); DINO-WM, V-JEPA 2 and LeWM plan with CEM in JEPA-style latent spaces (Zhou et al., 2025; Assran et al., 2025; Maes et al., 2026). iCEM improves CEM's sample efficiency with coloured noise and elite reuse (Pinneri et al., 2020), and DecentCEM runs several CEM instances with separate elites and keeps the best (Zhang et al., 2022). These methods treat one world model's cost as the objective; our change enters only through the cost, so it composes with any of them.

Ensembles and model exploitation. Ensembles of dynamics models capture epistemic uncertainty and penalize model exploitation in model-based RL (Chua et al., 2018; Kurutach et al., 2018; Janner et al., 2019; Yu et al., 2020; Kidambi et al., 2020), typically by propagating all candidates through all members; deep ensembles often give better uncertainty estimates than MC dropout (Lakshminarayanan et al., 2017; Gal & Ghahramani, 2016), and our error-correlation measurement gives a planning-specific reason: dropout copies share the planner's ranking errors. EliteCert uses independent models only to decide the elite boundary, which keeps the extra cost to a fraction of the planner's rollouts, and applies to pretrained models without retraining.

Screening, verification and top-k identification. A cheap model that screens and an expensive one that verifies is the pattern of detection cascades, cascade rankers in retrieval, multi-fidelity optimization and speculative decoding (Viola & Jones, 2001; Wang et al., 2011; Kandasamy et al., 2016; Leviathan et al., 2023); here the verifier need only be wrong in different places, and is spent only where the screen is undecided. Choosing the elites is a top-k identification problem, and the band is closest to racing and successive elimination, which keep sampling exactly the arms whose confidence intervals overlap the boundary (Maron & Moore, 1993; Even-Dar et al., 2006; Kalyanakrishnan et al., 2012; Bubeck et al., 2013). B_δ is the one-shot analogue of that overlap set, and Proposition 2 is the corresponding deterministic statement. Three things differ: the evaluations are deterministic but biased, so repeated queries of one model buy nothing and the second opinion must come from a different model; the width is calibrated on measured promotion gaps in the manner of split conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2023), applied here to an optimizer's elite boundary rather than to predicted trajectories or safety regions (Lindemann et al., 2023); and the allocation is one-shot.

Offline evaluation and the closed loop. Logged-data accuracy need not track control performance (objective mismatch, Lambert et al., 2020), and a learner judged on another policy's distribution can fail on its own (Ross et al., 2011); Proposition 1 is the form this takes when the scorer shifts the planner's candidate pools within one decision.

9 Conclusion

A world model can only make the decisions it can resolve. We made this precise with a decision certificate for population-based planners, measured it by executing every candidate a planner evaluates, and found that late in the search most elite decisions lie beyond the model's resolution. EliteCert acts on exactly those decisions: it spends independently trained models on a calibrated band around the elite boundary, runs unmodified CEM, iCEM and MPPI, and raises their success by 9–14 points on Push-T; with CEM it beats the default configuration at 44% of its CPU time. The same certificate explains why a better predictor helps, why correlated copies of the planner do not, and why gains on logged candidate pools cannot replace closed-loop evaluation.

Limitations and future work. EliteCert needs an additional model with partly independent errors. On Push-T predictors retrained on the planner's frozen encoder provide it; on Cube the gain comes from a model with its own encoder; on Reacher the qualification gate withholds the weaker models we could train. Gains are therefore largest where several comparable validators exist. Our evaluation uses LeWM models in simulation. Validators from a second model family such as DINO-WM, validators that also propose candidates beyond the band, and fused GPU kernels for the validator calls (currently 2.38 against 0.66 s per decision on a laptop GPU) are natural next steps.

REPRODUCIBILITY STATEMENT

All planners, environments and task protocols are those of the public `stable-worldmodel` library (the exact version is pinned in the supplementary code); the official-protocol runs reproduce its `eval_wm.py` task sampling (seed 42) with 600 tasks and planner seeds 42, 7 and 13, and our planner re-implementation reproduces the library's actions bit-for-bit when no validator is used (Section 2). All planning models and public validators are released checkpoints identified by repository and revision in Appendix F; the retrained predictors are trained on frozen encoder latents with the recipe, seeds and held-out split of Appendix I. The band width is computed from logged calibration pools by the procedure of Section 5, the qualification test uses fixed windows and perturbation seeds (Section 5, Appendix F), and the certificate counts, error correlations and proofs are in Appendices A and B. Every comparison is paired by task, with the tests stated where it is reported. An anonymized copy of the code, with a README mapping each script to the tables and numbers it produces, is provided as supplementary material and will be released.

USE OF LARGE LANGUAGE MODELS

Large language models were used in four ways: to proofread and polish the writing; to help write the programs that reproduce the experiments; to double-check the code, the reported numbers against the logged results, and the references against their published sources; and to search the literature for related work, which broadened the related-work discussion. The models were Claude Opus 5.5 (Anthropic), used through the Claude Code command-line agent, and GPT-5.6-sol (OpenAI), used through the Codex command-line agent. All experimental designs, results and claims were checked by the authors, who take full responsibility for the content.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Mahmoud Assran et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Sébastien Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple identifications in multi-armed bandits. In *International Conference on Machine Learning*, 2013.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 2018.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *International Conference on Machine Learning*, 2012.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B. Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control (L4DC)*, volume 120, pp. 761–770, 2020.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2023.

Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J. Pappas. Safe planning in dynamic environments using conformal prediction. *IEEE Robotics and Automation Letters*, 8(8): 5116–5123, 2023.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems*, 1993.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning (CoRL)*, 2020.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pp. 627–635, 2011.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer, 2004.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2017.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

Zichen Zhang, Jun Jin, Martin Jagersand, Jun Luo, and Dale Schuurmans. A simple decentralized cross-entropy method. In *Advances in Neural Information Processing Systems*, 2022.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, 2025.

A PROOFS, ERROR CORRELATION AND SCREEN RECALL

Proof of Theorem 1. Write e_k for the latent error after k steps. The one-step defect and the Lipschitz property give $e_{k+1} \le \kappa + Le_k$ with $e_0 = 0$, so $e_H \le \kappa \sum_{j<H} L^j$, and the read-out adds at most $\eta + \ell e_H = \varepsilon_H$. For two candidates with $Q_i - Q_j > 2\varepsilon_H$, $\hat{Q}_i \ge Q_i - \varepsilon_H > Q_j + \varepsilon_H \ge \hat{Q}_j$. For membership, let $q_{(k)}$ be the k-th largest true value. If $Q_i > q_{(K+1)} + 2\varepsilon_H$, every j with $\hat{Q}_j > \hat{Q}_i$ has $Q_j > Q_i - 2\varepsilon_H > q_{(K+1)}$, so it is one of the true top K other than i; fewer than K candidates outrank i and it is in the model's top K. If $Q_i < q_{(K)} - 2\varepsilon_H$, each of the true top K has $Q_j \ge q_{(K)} > Q_i + 2\varepsilon_H$ and hence $\hat{Q}_j > \hat{Q}_i$, so i is outside the model's top K. The model-side statement is the same argument with Q and $\hat{Q}$ exchanged, since the bound $|\hat{Q} - Q| \le \varepsilon_H$ is symmetric: if $\hat{Q}_i$ exceeds the model's $(K{+}1)$-th best value by more than $2\varepsilon_H$, every j with $Q_j > Q_i$ has $\hat{Q}_j > \hat{Q}_i - 2\varepsilon_H$ and is one of the model's top K other than i, so fewer than K candidates are truly better than i; and if $\hat{Q}_i$ falls short of the model's K-th best by more than $2\varepsilon_H$, each of the model's top K has $Q_j > \hat{Q}_i + \varepsilon_H \ge Q_i$. The factor 2 is tight: true values $\pm\varepsilon_H$ can both be estimated as 0. □

Proof of Proposition 1. Consider a deterministic two-iteration planner over scalar actions with true cost $J(a) = (a-1)^2$ (lower is better) and one elite per iteration: iteration 1 scores the pool $\{-2, -1, 0, 1, 2\}$ and keeps the minimizer μ_1; iteration 2 scores $\{\mu_1 - 1.5,\ \mu_1,\ \mu_1 + 1.5\}$ and executes its minimizer. The planning model scores $C_0(a) = (a-2)^2$, so the native planner keeps 2 and then executes 2 from $\{0.5, 2, 3.5\}$, at true cost 1. The logged native pools are these two sets. *Better offline, worse closed loop.* Let $S(a) = J(a)$ except $S(-0.5) = -1$, a point no logged pool contains. On the logged pools S selects 1 and 0.5 (true costs 0 and 0.25 against the native 1 and 1), strictly better on both. Driving the planner, S keeps 1, scores $\{-0.5, 1, 2.5\}$ and executes -0.5, at true cost $2.25 > 1$. *Worse offline, better closed loop.* Let $S(a) = J(a)$ except $S(0) = S(1) = S(2) = 10$. On the first logged pool S selects -1 (true cost 4 against the native 1), strictly worse. Driving the planner, S keeps -1, scores $\{-2.5, -1, 0.5\}$ and executes 0.5, at true cost $0.25 < 1$. □

The mechanism in both witnesses is the one in the main text: the scorer is evaluated offline on pools it did not generate, and in closed loop it generates its own.

Proof of Proposition 2. Let J be the planner's K lowest-cost candidates, so $z_j \le z_{(K)}$ for $j \in J$. For every $j \in J$, $s_j \le az_j + b + \varepsilon \le az_{(K)} + b + \varepsilon$. For a candidate i with $z_i > z_{(K)} + 2\varepsilon/a$, $s_i \ge az_i + b - \varepsilon > az_{(K)} + b + \varepsilon$. Hence K candidates have strictly smaller s than i, and i is not among the K smallest. The K smallest values of s therefore all lie in $B_{2\varepsilon/a}$, and selecting them inside $B_{2\varepsilon/a}$ or among all candidates gives the same set. □

With the deployed score, $s_i = \frac{1}{M+1}\big(z^B(c)_i + \sum_m z^B(c^m)_i\big)$ where z^B standardizes over the band; since $z^B(c)$ is an affine function of the pool-standardized z with positive slope, a and b absorb the planner's term exactly (fitted residual 9×10^{-15}) and ε bounds the validators' contribution. We checked the implication on 1028 held-out pools with no violation.

The condition is on the *shifted* disagreement, so a validator that is uniformly more or less pessimistic than the planner costs nothing; only disagreement about the order matters. The bound is also where the band differs from a fixed count: $|B_{2\varepsilon/a}|$ grows exactly when many candidates crowd the boundary, which is when a count is too small (Section 5).

Error correlation. Data: the full ground-truth audit behind Table 3 (native CEM $300{\times}30$, $K{=}30$, 561 decisions from 400 episodes, iterations $0, 5, 10, 15, 20, 29$, every candidate executed; 3361 pools after dropping five with constant outcome), of which the 70 decisions of Table 3 are a subset. For model k with in-pool standardized cost z^k and in-pool standardized true margin t, the residual $r^k = z^k - \rho_k t$ (ρ_k the pool correlation of z^k with t) is the part of the model's cost not explained by the outcome, and the difference error of a pair is $r^k_i - r^k_j$. We report the correlation of difference errors between models over all pairs of every pool, pooled within an iteration bucket (early: $0, 5$; middle: $10, 15$; late: $20, 29$), with 95% intervals from a bootstrap over episodes.

Difference-error correlation	early	middle	late
planner vs. validators (561 decisions)	0.79	0.70	0.54
validator vs. validator	0.82	0.76	0.66
planner vs. validators (70 decisions, it. 5–29)	0.74	0.65	0.51
planner vs. MC dropout (same 70 decisions)	0.93	0.88	0.71

Intervals are within ± 0.03 on the 561 decisions and within ± 0.09 on the 70; the late-iteration intervals for validators ($[0.42, 0.59]$) and dropout ($[0.63, 0.77]$) do not overlap. Using ranks instead of standardized costs gives the same ordering (0.75, 0.63, 0.44 for planner vs. validators). The validators' own error variance is 11–19% below the planner's, consistent with their better qualification scores.

Screen recall. On the same pools, the fraction of the true top 30 (by executed outcome) that each screen contains, early / middle / late: planner top K $0.38/0.18/0.10$; planner top $2K$ $0.54/0.32/0.19$; band δ=0.48 $0.66/0.63/0.50$ at mean sizes $91/143/152$ of 300. At matched size the band recalls no more of the true top K than the planner's top $2K$ (δ=0.1, about 60 candidates: $0.46/0.32/0.20$). The band is therefore not a better screen for the environment's elites; what it covers is the set certification would promote ($0.89/0.89/0.79$ of promotions at δ=0.48, against $0.71/0.50/0.31$ for the top $2K$). Candidates that are good in the environment but outside the band are invisible to every scoring rule applied inside it, which is a limit of any screen-then-verify planner and a reason to let independent models also *propose* candidates, which we leave to future work.

B CERTIFIED DECISIONS BY SCORER AND ITERATION

Figure 2 plots the 70-decision audit of Table 3: the planner's rank correlation with the executed outcome falls over the search while the elites chosen with three independent models stay better than the planner's own, and the table below extends the count of certified decisions to the full 561-decision audit.

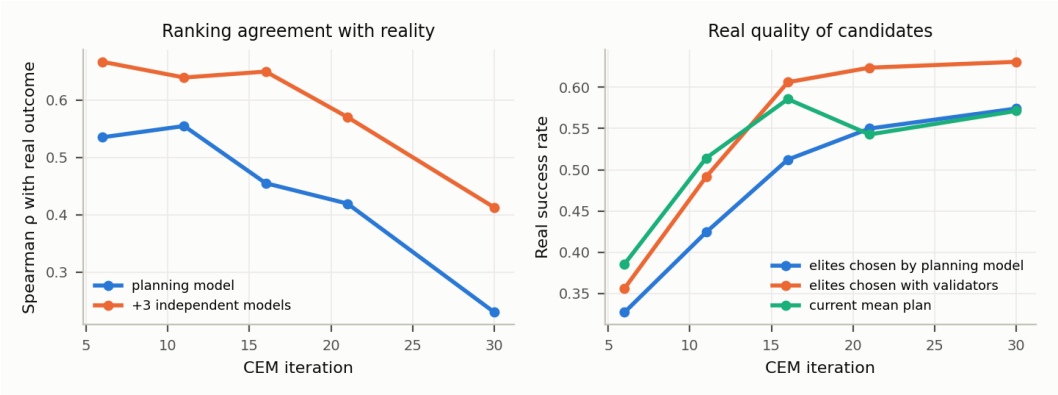


Figure 2: Ground-truth audit on Push-T (70 decisions, every candidate simulated). Left: agreement between cost ranking and real outcome over the 300 candidates of an iteration. Right: real success rate of the 30 elites chosen by the planning model alone, of the elites chosen when the cost is averaged with three independently trained models, and of the planner's current mean plan.

Table 3: Ground-truth audit of native CEM on Push-T (70 decisions). ρ: Spearman correlation between the model's cost and the real outcome margin over the 300 candidates of an iteration. Success rates are measured by executing plans in the simulator.

Iteration	ρ(model, real)	real success of mean plan	of model elites	of random candidates
0	0.74	1%	9%	4%
5	0.54	39%	33%	14%
10	0.56	51%	42%	21%
15	0.46	59%	51%	29%
20	0.42	54%	55%	36%
29	0.23	57%	57%	50%

Full audit, 561 decisions, N=300, K=30. For each pool and scorer, the standardized cost is regressed on the standardized true margin; ε is the median absolute residual pooled within early (iterations 0, 5), middle (10, 15) and late (20, 29) phases. Certified: $|z_i - z_{(K)}| > 2\varepsilon$. Error: fraction of certified in/out labels that disagree with true top-K membership. Means are planner plus three models; dropout rows use the 70 decisions where it was logged.

Certified fraction (error)	it. 0	5	10	15	20	29
planner	.67 (.04)	.46 (.06)	.35 (.06)	.32 (.07)	.28 (.09)	.29 (.12)
+ 3 public models	.71 (.04)	.51 (.04)	.42 (.04)	.39 (.05)	.37 (.06)	.37 (.07)
retrained predictor (one)	.68 (.04)	.50 (.05)	.42 (.05)	.40 (.05)	.39 (.05)	.43 (.06)
+ 3 retrained predictors	.69 (.04)	.49 (.05)	.41 (.04)	.39 (.05)	.38 (.05)	.38 (.06)
+ 3 recurrent predictors	.70 (.04)	.48 (.05)	.41 (.04)	.40 (.05)	.39 (.05)	.39 (.06)
+ 3 MC-dropout copies	—	.45 (.05)	.38 (.06)	.32 (.06)	.31 (.08)	.28 (.10)

The empirical ε (early/middle/late) is $0.43/0.47/0.49$ for the planner, $0.38/0.39/0.38$ with the public models, $0.42/0.43/0.43$ for a single retrained predictor and $0.50/0.45/0.43$ with dropout copies. With the 90th-percentile residual as ε the same ordering holds at lower fractions (planner $0.14 \to 0.08$). One-step held-out errors: official 0.0079, retrained transformer predictors 0.0045, recurrent predictors 0.0081.

C METHODS THAT DID NOT HELP

All numbers below are Push-T, CEM, and the same 200 tasks unless stated otherwise.

Truncated-rollout elimination. After rolling all candidates to depth $d < H$, we bound each candidate's final cost with a per-depth split-conformal interval around a regression on its prefix costs (fitted on half of the logged decisions, evaluated on the other half) and discard candidates whose lower bound exceeds the K-th smallest upper bound. We tried global, per-iteration demeaned, and spread-normalized residual models, schedules $\{1,5\}, \{2,5\}, \{3,5\}, \{1,2,5\}, \{1,3,5\}, \{1,2,3,4,5\}$ and risk levels $\alpha \in \{0.1, 0.05, 0.01, 0.001\}$. The largest saving at a violation rate below 1% was 0.2% of rollouts, against 72% for an oracle that knows the elites. The reason is visible in the raw data: the Spearman correlation between depth-4 and depth-5 cost falls from 0.88 at the first CEM iteration to 0.52 at iteration 10 and 0.14 at the last.

Relational smoothing. Selecting elites by a Gaussian-kernel average of costs over candidates in action space (bandwidth 0.25–$1\times$ the median pairwise distance), optionally including remembered elites of earlier iterations, reduced success at every budget (RTX 4060 runs, each against its own native baseline), e.g. 64×5: 58.5% native, 56.5% (0.25), 36.0% (0.5), 30.0% (1.0); 150×10: 71.5% native, 32.5% (0.5), 44.5% (0.5 with memory). Within an iteration a candidate's cost is poorly predicted by its neighbours (Spearman 0.1–0.2 after the first iteration), yet this idiosyncratic component is real: removing it removes the planner's signal.

Adaptive stopping. Using the logged decisions and the ground-truth outcomes of the mean plan at iterations $\{1, 2, 3, 5, 8, 10, 15, 20, 25, 30\}$, we compared fixed budgets with stopping when the elite standard deviation falls below a threshold, when the planner's cost of the mean plan stops improving

by more than δ, and when the validators' cost of the mean plan stops improving. No rule exceeded the real success of the fixed $T{=}15$ budget (58.6%) at a smaller average number of iterations; choosing among iterates by validator cost reached 60.0%. Closed loop, stopping at elite spread < 0.75 reached 80% success with 60% of the iterations, against 82% for a fixed $T{=}15$.

Symmetry-shifted copies of the planning model. A cheaper way to manufacture a second opinion is to score each candidate again under small shifts of the observation — here the planning model itself re-run on ± 3-pixel translations in each axis, its four scores averaged with the unshifted one. Closed-loop over 200 official tasks this changes success by 0.0 points on Cube (65.5 both ways, five tasks won and five lost) and -7.5 on Reacher ($68.5 \rightarrow 61.0$, 25 won against 40 lost, $p{=}0.08$). The augmented scores are more stable than the original but wrong in the same places, which is the same failure as MC dropout below, measured in the loop rather than on logged pools.

MC-dropout validators. Three dropout-enabled forward passes of the planning model's predictor used as validators changed neither the rank correlation with the real outcome (0.46 vs. 0.49 at iteration 15) nor the real success of the chosen elites (51.2% vs. 51.2%), while three independently trained models raise the latter to 60.6%.

D BAND CALIBRATION IN FULL

Table 4 calibrates δ to nominal coverage targets and measures the coverage obtained on native and on certified pools; Table 5 gives closed-loop success and timing across δ on GPU and CPU. Section 5 and Section 6.5 discuss them. The shortfall of measured against nominal coverage comes from gaps clustering within pools; stratifying the quantile by iteration does not remove it (0.877 against 0.874 at the 90% target).

Table 4: Calibrating the band to a coverage target, at the deployed configuration (Push-T, CEM $150{\times}10$, $K{=}15$, three validators). δ is the $(1-\alpha)$ quantile of the promotion gap on calibration pools from the native planner; coverage and band size are measured on test pools from disjoint episodes, and separately on pools logged from a certified run — the distribution the band is actually used on. 1348 native and 396 certified pools.

		measured coverage		band $\lvert B_\delta \rvert / N$
nominal $1-\alpha$	calibrated δ	native pools	certified pools	native / certified
0.80	0.296	0.784	0.817	0.26 / 0.25
0.90	0.481	0.867	0.925	0.35 / 0.34
0.95	0.686	0.914	0.964	0.43 / 0.43
0.99	1.260	0.973	0.992	0.63 / 0.62

Table 5: Resolution band calibration. 200 Push-T tasks $\times$ 3 planner seeds per cell, three validators. GPU: $300{\times}10$ on an RTX 4060; CPU: $150{\times}10$ on a Mac Studio, one process at a time. Success is flat in δ on both.

δ	GPU success	GPU median s	CPU success	CPU median s
0.1	82.8	2.24	82.8	1.49
0.25	84.0	2.22	83.0	2.87
0.5	**86.1**†	2.23	84.5	3.10
1.0	84.3	2.22	**85.0**	2.04
native	74.5	0.67	75.2	1.32

†six seeds.

E OFFLINE GAINS THAT DID NOT REACH THE LOOP

Five times in this work an improvement that was clear on logged data or candidate pools did not survive closed-loop evaluation, and in the same direction each time; we number them as in Section 7.

(i) A certifier supervised on real outcomes improved true elite quality at late iterations from 0.567 to 0.625 offline and lost 22.5 points closed-loop when it replaced the planner's cost. A learned head on the planner's own predicted path, including the order information in its level-2 path signature, moved offline elite regret by 0.0%; it did not improve offline and is not counted among the five. (ii) Dropping the planning model's cost from the certified score — it has already spoken by choosing the band — raised offline elite quality by 3.7 points monotonically in its weight, and changed closed-loop success by $+1.0$ and -1.2 points at two band widths, neither significant over three seeds. (iii) A third attempt changed the dynamics rather than the score: a 33k-parameter residual layer that carries a memory of real observation innovations and corrects the predicted latent, fitted on dataset trajectories and driven at test time by the observations the agent actually receives. Offline it raised true elite quality at *every* CEM iteration (0.543 to 0.570 at the last), and it did so by genuinely reordering candidates: 9 to 44 per cent of the correction varies across candidates and elite overlap with the uncorrected ranking falls to 0.37. Closed-loop, against the stock planner over 200 tasks and three seeds, it is -0.2 points (43 wins, 44 losses, p=1.00), with a latent-history control confirming that the comparison isolates the memory. That attempt had none of the escape hatches of the first two — its correction reorders, the reordering is right offline, and its state is driven by real observations rather than by planner candidates – and it still transferred nothing. (iv) Replacing its geometry with a plain recurrent cell of the same budget sharpens the point: that control raises offline elite quality 2.5 times as much (0.543 to 0.612 at the last iteration) and is 2.5 points *worse* closed-loop, negative on all three seeds. Between the two correctors the offline and closed-loop orderings are inverted.

(v) That run, however, tested less than the proposal. The layer's read/write geometry was left unfitted — a random orthonormal subspace and an identity error metric — its windows used a three-block horizon where the planner rolls five, and its checkpoint was chosen on the trajectories the result was then read from. Repeated with the geometry estimated by ridge regression and eigendecomposition of the residual signal-to-noise on two disjoint partitions of training episodes, windows matching the five-block decision at warmups of 0, 2 and 4 blocks, and checkpoints chosen on held-out validation episodes, the layer improves five-step latent error by 14.0% against the frozen model, against 12.4% for the same layer with a random geometry and 11.1% for a budget-matched recurrent cell. Closed-loop over 200 tasks and three planner seeds it reaches 71.2% against 75.2% native (-4.0, p=0.03), 75.2% for the recurrent control (-4.0, p=0.02) and 74.2% for the random geometry (-3.0, p=0.07), at $3.4\times$ the planning time. Fitting the geometry and aligning the training to deployment improved every offline number and made the closed-loop result worse, and the offline ordering of the three is again the reverse of the closed-loop one.

The common cause we can identify is that a scorer replayed on logged pools is judged on a candidate distribution the native planner created, while in deployment it creates its own; but after five instances we would put it more plainly. On this task we have found no dependable relation between offline elite quality measured on logged pools and closed-loop success, and where we can compare two methods on both, the relation runs backwards. A re-ranking result reported only on logged pools is not evidence that the planner improves, so we report none of ours on offline pools alone: fixed-pool replay is a screening step, not evidence. The one offline quantity we do report, the band's coverage, is a property of the screening step rather than a prediction about success, and we still checked it on pools from a certified run, where it holds conservatively.

F Implementation and evaluation details

Qualification test: 600 dataset windows drawn with seed 123 (from held-out episodes for the re-trained predictors), each executed action sequence scored against 31 Gaussian perturbations at σ=0.1 in normalized action units. Planning model: the official LeWM Push-T checkpoint; validators: three publicly released LeWM checkpoints trained with different seeds on the same Push-T data (`sadiash/lewm-sprint2-pusht`). Solver: the `stable-worldmodel` CEM, MPPI (temperature 0.5, softmax over the K elites) and ICEMSolver (library defaults), horizon 5 blocks of 5 actions, receding horizon 5, $K = N/10$. Push-T tasks start at step 15 of held-out dataset episodes with the goal 25 steps later and a 50-step budget; success is the environment's test (position error < 20, angle error $< 20°$). The official-protocol runs reproduce `eval_wm.py` of `stable-worldmodel`: start rows sampled with `default_rng(42)` among dataset rows with at least 26 remaining steps, goal = dataset frame 25 steps later, 50-step budget, environment-specific state-setting callables, success from `World.evaluate_from_dataset`; we draw 200 (Push-T and Cube: 600) instead

of 50 tasks and plan each task in its own environment. Reacher uses the `qpos_match` task. Cube validator: `cgpadwick2020/lewm-cube`; planner `FadyRezk/lewm-cube`. Reacher validators: three models of the same architecture that we trained on the Reacher dataset for 10 epochs (different seeds). Timing: median over all planning decisions of 50 tasks (40 for Table 2), one configuration at a time, encodings computed once per decision for every arm.

G TIMING AND THE PLANNER-IDENTITY CONTROL

A note on measuring the overhead. Certification overhead is easy to overstate: running the validators as one stacked vectorized model, correct on GPU, is 39% slower on CPU because `vmap` falls back to a per-sample loop for attention, and process contention inflates the validator arms further. All CPU timings here are single-process with the device-appropriate path.

Every CEM budget. Table 10 lists four CEM budgets; three more were run on the same 200 tasks with one validator. Native against EliteCert, in rollouts per decision: 64×5 (320) $57.5 \to 62.5$ (p=0.16); 100×5 (500) $56.5 \to 66.5$ (p=0.002); 64×10 (640) $66.5 \to 75.5$ (p=0.005); 150×5 (750) $60.0 \to 64.0$ (p=0.27); 150×10 (1500) $74.5 \to 83.0$ (p=0.005); 150×15 (2250) $76.0 \to 81.5$ (p=0.06); 300×10 (3000) $70.0 \to 79.5$ ($p=3\times10^{-4}$). The gain is positive at all seven, between 4.0 and 10.0 points, and resolved at four of them on 200 tasks.

The official comparison over planner seeds. Table 2 runs one planner seed. Both arms repeated on two more, over the same 600 official tasks (default: 300×30; EliteCert: band δ=0.481):

Seed	Default	EliteCert	Δ [95%]	w/l	p
42	85.67	92.00	+6.33 [+3.3, +9.3]	61/23	4×10^{-5}
7	88.67	90.00	+1.33 [−1.5, +4.0]	39/31	0.40
13	87.83	91.50	+3.67 [+0.7, +6.5]	50/28	0.017
pooled	87.39	91.17	+3.78 [+2.1, +5.4]	150/82	9×10^{-6}
Cube, native 150×10 *against the same band width*					
42	67.83	70.83	+3.00 [+0.8, +5.3]	33/15	0.013
7	65.83	69.00	+3.17 [+0.7, +5.7]	38/19	0.016
13	65.67	68.33	+2.67 [+0.2, +5.2]	37/21	0.048
pooled	66.44	69.39	+2.94 [+1.5, +4.5]	108/55	4×10^{-5}
Reacher, native 150×10 *against three validators that fail the qualification test*					
42	68.5	57.5	−11.0	24/46	0.012
7	65.0	63.0	−2.0	26/30	0.69
13	63.0	57.0	−6.0	26/38	0.17
pooled	65.5	59.2	−6.3	76/114	0.007

Seed 42, the one in Table 2, is the most favourable of the three, and the reason is the baseline rather than the method: the default planner ranges over 85.7–88.7 across seeds while the certified planner ranges over 90.0–92.0. The pooled $+3.8$ agrees with an independent three-seed comparison at 200 tasks, which gives $+3.8$ (p=0.014) for a fixed count against the same default. Cube behaves differently: the same band width, transferred unchanged from Push-T, gives $+3.0$, $+3.2$ and $+2.7$, every one resolved. Seed 42 happens to be the extreme on Push-T and Reacher — the largest gain on Push-T and the largest loss on Reacher — which is why the main text quotes pooled figures.

Table 6: Median planning time per decision (s) and success on Push-T. CPU: exclusive 16-core node; GPU: RTX 4060 laptop. Table 2 uses a single 8-thread process, so absolute times are not comparable across the two tables. M: validators; "late": validators only from iteration $T/2$.

Configuration	CPU time	GPU time	Success (CPU / GPU)
Native 150×10	1.23	0.68	74.5 / –
Native 150×15	1.89	0.99	76.0 / 76.0
Native 300×15	3.23	0.96	77.5 / 73.5
Native 300×30 (default)	7.61	1.90	77.5 / 79.5
EliteCert 150×10, M=1	1.91	1.36	83.0 / 79.5
EliteCert 300×10, M=1, late	–	1.01	– / 82.5

Table 7: Planner identity control, Push-T, CEM 150×10, 200 paired tasks.

Planner	Validators	Success	Δ vs. same planner alone (p)
official	–	74.5	
official	seed 0	83.0	+8.5 (0.005)
seed 0	–	77.0	
seed 0	official	80.0	+3.0 (0.35)
seed 0	seeds 1, 2	83.0	+6.0 (0.029)
seed 0	seeds 1, 2, official	83.5	+6.5 (0.041)
seed 1	–	79.0	
seed 1	seed 0	81.0	+2.0 (0.54)
seed 1	seeds 0, 2	84.5	+5.5 (0.052)
seed 1	seeds 0, 2, official	85.5	+6.5 (0.011)

H FULL RESULT TABLES AND THE COMPUTE FRONTIER

Table 8 gives the official-protocol results per environment with the offline qualification scores; Table 9 the three planners at a matched budget over six planner seeds; Table 10 the paired comparisons at every budget on the in-house protocol; Figure 3 the resulting success-versus-compute frontier.

Table 8: Official LeWM protocol, 200 tasks per environment (*600 tasks), CEM, one planner seed. $q_{0.1}$: offline qualification score of planner / validators (lower is better), from held-out dataset windows; a later repetition with bootstrap intervals moves every value by at most 0.01 and changes no admission decision. Gated: EliteCert with the qualification rule. †band with δ carried over from Push-T, not recalibrated.

Env	Config	$q_{0.1}$ planner / val.	Native	EliteCert	Δ (p)	Gated
Push-T*	150×10, M=1	0.37 / 0.27	82.5	89.3	+6.8 (< 0.001)	89.3
Push-T*	150×10, M=3	0.37 / 0.27–0.29	82.5	91.2–92.0	+8.7–9.5 ($< 10^{-8}$)	91.2–92.0
Cube*	150×10, M=1, count $2K$	0.35 / 0.17	67.8	70.5	+2.7 (0.023)	70.5
Cube*†	150×10, M=1	0.35 / 0.17	67.8	70.8	+3.0 (0.013)	70.8
Cube	300×10, M=1	0.35 / 0.17	63.5	65.5	+2.0 (0.50)	65.5
Reacher	150×10, M=3	0.34 / 0.35	68.5	57.5	−11.0 (0.012)	68.5
Reacher	300×10, M=3	0.34 / 0.35	62.5	54.0	−8.5 (0.033)	62.5

Table 9: Certification across planners. 200 Push-T tasks × 6 planner seeds, 300×10, GPU, band δ=0.5 with three validators. iCEM runs unmodified through the cost-model wrapper.

Planner	Update rule	Native	EliteCert	Δ (p)	rel.	median s
CEM	top-K truncation	74.6	86.1	+11.5 (5×10^{-25})	+15.4%	0.67 → 2.23
iCEM	truncation, elites carried over	37.6	51.2	+13.6 (9×10^{-20})	+36.1%	0.93 → 2.63
MPPI	exponential weights over elites	53.6	70.0	+16.4 (3×10^{-24})	+30.6%	0.95 → 2.35

Table 10: Paired comparisons on 200 Push-T tasks, *one planner seed*: the intervals and tests are paired over tasks only, so they do not carry the seed-to-seed variation, which is ± 2.5–4.5 points. Table 9 repeats all three planners at one budget over six seeds. Δ: success difference (percentage points) with 95% bootstrap interval; p: McNemar.

Planner	Config	Native	EliteCert	Δ [95% CI], p
CEM	64×10, $M{=}1$	66.5	75.5	$+9.0$ [$+3.0$, $+15.0$], 0.005
CEM	100×5, $M{=}1$	56.5	66.5	$+10.0$ [$+4.0$, $+16.0$], 0.002
CEM	150×10, $M{=}1$	74.5	83.0	$+8.5$ [$+3.0$, $+14.0$], 0.005
CEM	150×15, $M{=}1$	76.0	81.5	$+5.5$ [$+0.5$, $+10.5$], 0.06
MPPI	100×10, $M{=}3$	45.5	61.0	$+15.5$ [$+7.5$, $+23.5$], < 0.001
MPPI	150×10, $M{=}3$	54.0	66.5	$+12.5$ [$+5.0$, $+20.0$], 0.002
MPPI	300×15, $M{=}3$	48.5	65.0	$+16.5$ [$+9.5$, $+24.0$], < 0.001
iCEM	150×10, $M{=}3$	56.5	73.0	$+16.5$ [$+10.0$, $+23.0$], < 0.001
iCEM	300×15, $M{=}3$	34.0	52.0	$+18.0$ [$+10.5$, $+25.5$], < 0.001

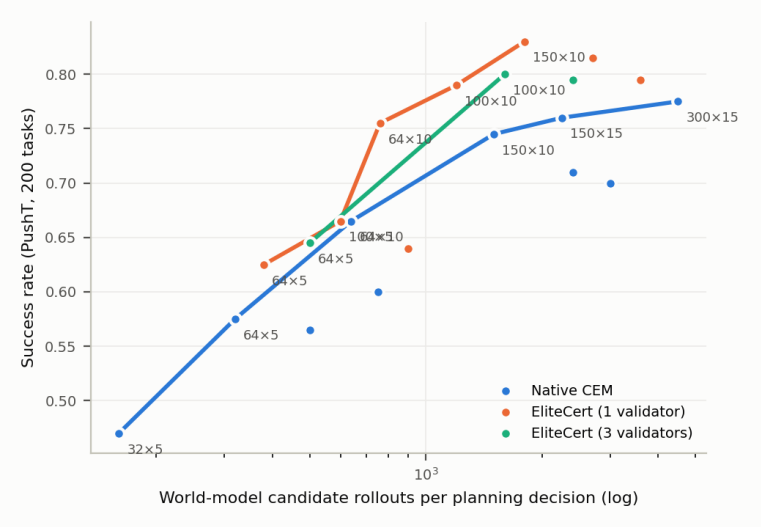


Figure 3: Push-T success versus world-model rollouts per decision (CEM, 200 tasks per point; labels give candidates×iterations). EliteCert with a single validator dominates the native frontier above a few hundred rollouts per decision.

I Validators from a retrained predictor

Pre-registered before any run: train three predictors (official architecture including the action encoder, seeds 0/1/2, fresh initialization) on the planning model's frozen encoder latents with the LeWM prediction loss (teacher forcing, three frames of history, frameskip 5), holding out 5% of episodes and choosing the epoch by held-out error only; admit them if $q_{0.1}$ on 600 held-out windows is at most the planner's (G1); then run the official protocol with M admitted predictors, CEM 150×10, $\delta{=}0.481$ (G2), extending to three planner seeds only if seed 42 is positive.

Env	held-out error new / official	$q_{0.1}$ planner / new	G1	closed loop, seed 42
Push-T	0.0045 / 0.0079	0.36 / 0.26–0.29	pass	$82.5 \to 91.3$ ($+8.8$, $p{=}2\times10^{-10}$)
Cube	0.016 / —	0.36 / 0.22–0.24	pass	$67.8 \to 67.8$ (0.0, $p{=}1.0$)
Cube-LN	0.0033 / 0.0049	0.276 / 0.225–0.228	pass	$67.17 \to 68.67$ ($+1.50$, $p{=}0.289$)
Reacher	0.023 / 0.028	0.34 / 0.34	pass (marginal)	$65.8 \to 67.0$ ($+1.2$, $p{=}0.66$)
TwoRoom-e30	0.0080–0.0091 / 0.0026	0.019 / 0.039–0.053	fail	not run

On Push-T all three seeds are positive ($+8.8$, $+10.7$, $+7.8$; pooled $+9.1$). The control with a retrained predictor as planner (not pre-registered) gives $89.2/89.8/89.0\%$ alone and $91.3/91.5/90.5\%$ with

certification. Cube is the first environment where the qualification test admits validators that do not help in the loop; the original encoder gives $67.83 \to 67.83\%$ for the retrained validators, while the corrected final-LayerNorm rerun gives $67.17 \to 68.67\%$ ($+1.50$ $[-1.0, +3.8]$, p=0.289). On TwoRoom, extending training to 30 epochs still fails qualification for all three seeds, so no closed-loop run is justified.

Does the better-predictor lever transfer? Registered before the runs: the retrained seed-0 predictor as planner on Cube (with the corrected encoder) and Reacher, alone and certified by the official model and the other two retrained predictors, 600 official tasks, planner seed 42. Planning with the retrained predictor gives $67.2 \to 67.5\%$ on Cube ($+0.3$ $[-2.3, +3.0]$, 34/32, p=0.90) and $65.8 \to 68.0\%$ on Reacher ($+2.2$ $[-2.3, +6.8]$, 105/92, p=0.39); certifying it gives 68.8% ($+1.3$ over it, p=0.30) and 67.7% (-0.3, p=0.94). No comparison meets the registered criterion, so no further seeds were run. A first submission of the planner-only arms omitted the retrained checkpoint and reproduced the official planner task for task; those runs were discarded.

Why the lever is large on Push-T and small elsewhere. Theorem 1 splits the value error into a predictor term $\ell\kappa \sum_{j<H} L^j$ and a read-out term η, the part that no predictor can remove because it lies in the encoder and in the latent-distance cost. Retraining the predictor on a frozen encoder can only shrink the first. The three environments differ in how much it shrinks and in which term dominates. On Push-T the retrained predictors lower held-out one-step error by 43% ($0.0079 \to 0.0045$) and improve the qualification score by 0.07–0.10, the largest change of the three, and success rises by 6.8 points. On Reacher the one-step error falls by only 18% ($0.028 \to 0.023$) and the qualification score does not move (0.34 both), so there is little to gain, and the $+2.2$-point change is consistent with that. Cube is the informative case: one-step error falls by 33% ($0.0049 \to 0.0033$) and the qualification score improves by 0.05, yet neither planning with the retrained predictor ($+0.3$) nor certifying with retrained predictors on the shared encoder ($+1.5$, $+1.3$) resolves a gain, while the one public validator with its own encoder adds 2.9 points ($p = 4 \times 10^{-5}$). This is what the theorem predicts if Cube's errors sit mainly in the read-out term: better dynamics on a shared encoder leave them in place, and only a model that encodes the scene differently can disagree with them. It is also a further instance of Proposition 1: a lower dataset error and a better offline qualification score did not certify a closed-loop gain. We then tested this reading directly, with a prediction registered before any Cube ground truth was collected: execute every candidate of 60 Cube decisions (CEM 150×10, iterations 0/3/6/9, 36,000 simulated plans; margin = final cube-to-goal distance over the 0.04 m success radius) and compare the planner's difference errors with those of the three retrained predictors and of the public model. In the late iterations the retrained predictors' errors correlate with the planner's at 0.85 $[0.80, 0.89]$, the public model's at 0.58 $[0.48, 0.67]$ (51 decisions with exact executed-plan replay; 0.85 and 0.58 on all 60), and adding the public model raises the certified fraction by 0.052 against 0.013 for the retrained ensemble. Predictors retrained on the shared encoder thus inherit the planner's errors on Cube and a model with its own encoder does not, as the prediction required. One qualification limits how far this goes: on Cube every model ranks candidates only weakly against the true margin (Spearman 0.07–0.12, against 0.23–0.74 on Push-T), so the certified fractions are small effects on a weak signal, and the correlation contrast is the more informative of the two numbers.

Architectural-diversity follow-up. We write AR0–2 for the three retrained predictors with the official autoregressive (AR) transformer architecture and GRU0–2 for three gated-recurrent-unit (GRU) predictors of matched parameter count, all trained on the same frozen Push-T encoder latents and loaded strictly alongside the official encoder. On held-out windows GRU0–2 have $q_{0.1} = 0.325/0.349/0.322$ and AR0–2 $0.256/0.285/0.269$, against 0.359 for the official planner, so all six pass the official-planner gate. The official planner with GRU0/1/2 reaches 88.83% on 600 tasks versus 82.50% native ($+6.33$ $[+3.3, +9.2]$, 60/22, $p = 3.2 \times 10^{-5}$); AR-retrained validators reach 91.33% on the same tasks (-2.50 $[-4.8, -0.3]$ for GRU versus AR, 17/32, p=0.044). A four-arm stress test with an AR planner gave: AR+GRU 89.33%, AR+AR1/2+GRU0 91.33%, GRU native 84.33%, and GRU+AR 90.17%. These arms are diagnostic because the AR planner is stronger than the qualification reference; the mixed arm is not used as a gated headline.

GPU equivalence audit. On an RTX 4060 Laptop, the eager reference medians were 0.6605 s (native) and 2.3788 s (certified). Compilation of the certified path failed in the tested backend. A

CUDA-graph path reached 0.6807 s for certification but matched the eager action bit-for-bit on only 95.45% of replay decisions (elite sets on 98.48%), so it is a non-equivalent implementation rather than a certified speedup. The full fused path was 1.5254 s but changed actions on roughly 30% of decisions. We therefore make no GPU speedup claim for these variants.

Mixing the two architectures. The official planner with AR1/AR2/GRU0 reaches 89.83% on 600 tasks at seed 42. Against pure AR validators the change is -1.50 $[-3.33, +0.33]$, 11/20, p=0.150; against pure GRU validators it is $+1.00$ $[-1.17, +3.17]$, 25/19, p=0.451; against public validators it is -2.17 $[-4.33, 0.00]$, 15/28, p=0.066. An additional run on the identical task set gives 90.83% at planner seed 7 and 91.67% at seed 13. Pooled over seeds 42/7/13, mixed minus pure AR is -0.83 $[-1.94, +0.28]$ points with task-cluster bootstrap, 41/56, descriptive McNemar p=0.155; the extension was selected after observing seed 42 and is not treated as a prospective test. This closes the qualified architecture-mixing comparison without supporting a mixed-validator improvement claim.

J Ablations of EliteCert

Table 11: EliteCert variants, CEM 150×10, 200 Push-T tasks, one planner seed (native: 74.5%). M validators, boundary rK.

Variant	Success	Extra rollouts / iteration
M=1 (seed 0), r=2	83.0	30
M=1 (seed 1), r=2	78.5	30
M=1 (seed 2), r=2	80.0	30
M=2, r=2	82.5	60
M=3, r=1.5	82.0	68
M=3, r=2	79.5	90
M=3, r=3	83.5	135
M=3, all candidates	85.5	450
M=1, validators from iteration 5	79.0	15
M=3, validators from iteration 5	83.5	45

Planner seed. With a second planner seed (7 instead of 42) on the same 200 Push-T tasks (150×10), native CEM reaches 72.0%, EliteCert 78.5% with one validator ($+6.5$, p=0.024) and 83.0% with three ($+11.0$, $p < 0.001$); the default 300×30 configuration reaches 79.5%.

How the elites become a distribution. Certification decides which candidates are elites; whether the hard top-K cut is the right way to turn them into the next sampling distribution is a separate question. We compared it with two alternatives that use only the scores already computed on the band, so neither costs a model call: weighting every candidate by its probability of being in the top K under the models' disagreement, and projecting the full-ensemble update together with the M leave-one-model-out updates onto a single Gaussian by moment matching. Over 200 official tasks and three planner seeds at δ=0.481 the three reach 89.7, 89.2 and 89.0%: within noise of each other (15 to 19 discordant units out of 600) and at the same wall-clock. That is not because the rules agree. On 1348 logged pools the leave-one-model-out hypotheses select the same elite set in only 2.5% of iterations, and where they differ about four of the fifteen elites are contested. The rule can therefore be changed on almost every iteration, moving a quarter of the elite set, without the closed loop noticing: what the planner is sensitive to is which candidates get certified, not how the elites are averaged afterwards.

Is the calibrated band padded? A cheaper band would be worth having if the candidates it drops did not matter to the update. Because the update is a diagonal Gaussian fitted to the elites, it depends on them only through their mean and per-coordinate standard deviation, and two different elite sets can produce the same one. We therefore measured, on 1348 cached pools from 240 episodes, how far narrowing the band moves the next sampling distribution — the displacement $\|\Delta\mu\|^2 + \frac{N-1}{N}\|\Delta\sigma\|^2$ of the next pool, in units of the current sampling scale. Narrowing δ to 0.40 in the early iterations saves 4% of the validator rollouts and already moves the update by more than a tenth of a sampling standard deviation on 36% of episodes; reaching a quarter of the rollouts costs 0.37. The band is not

padded in this sense: its members carry the moments the update is made of, and elite overlap stays above 0.98 while the displacement is already large, so overlap is the weaker diagnostic. A staged variant that runs one validator on the whole band and the others only where Proposition 2's bound leaves membership open fares no better: on 632 held-out pools it avoids at most 5% of validator rollouts, and only with a bound that holds on under a third of pools; with a bound wide enough to hold, it saves under 1.5%. Almost no candidate inside the band can be decided early, which is what the band is for. Stopping after the first validator instead, whenever the two-model elite boundary is separated by more than a threshold τ times their largest disagreement, fares no better on the same held-out pools: no τ keeps the elite set equal to the three-validator one on 95% of stopped pools while saving 20% of validator rollouts (τ=0.1 saves 3.4% and changes 62% of the stopped elite sets), so we did not run it in the loop.

That displacement does not, however, predict success. At δ=0.1 the update is displaced past a tenth of a standard deviation on 98% of episodes and by 0.32 at the 95th percentile, and closed-loop success there is 85.5% — the same as certifying every candidate (Table 11). Together with the elite-set result above, this says the planner tolerates large changes to both the elite set and the distribution fitted to it, which is why δ is calibrated against a coverage target and then checked against closed-loop success rather than against update fidelity.

Could one band width be chosen per task? Success at δ=0.1 and δ=1.0 differs by three points in the mean (82.5 against 85.5%), but the two do not succeed on the same tasks: over 600 (seed, task) units the wide band wins 34 that the narrow one loses and the narrow band wins 16 that the wide one loses. Choosing per task with every label known would give 87.5% against 85.5%, so a perfect oracle is worth two points. It is not reachable, because the preference is a property of the run and not of the task: carrying the true per-task preference from one planner seed to the others changes success by -0.5, 0.0, 0.0 and -0.25 points across the four splits, and on the tasks one seed marks as suiting the narrow band the held-out seeds give the wide band 92.9% against 78.6%. The discordant units are realizations of (task, seed). This is worth stating because the same pattern would defeat any per-task selection rule, however it is fitted.

One validator on the other planners. Table 10 certifies MPPI and iCEM with three validators, but most environments have only one other public checkpoint, as Cube does. Repeating those configurations with a single validator, over the same 200 tasks: MPPI gains $+14.5$ points at 300×15 ($48.5 \rightarrow 63.0$, p=5×10^{-4}) and $+5.5$ at 150×10 (p=0.19), and iCEM $+11.5$ at 150×10 ($56.5 \rightarrow 68.0$, p=0.004) and $+11.0$ at 300×15 (p=0.003). Against the three-validator rows of Table 10 that is most of the gain at 300×15 and roughly half of it for MPPI at 150×10: the second opinion is what matters, and further validators sharpen it.