Scaling Quadruped World Models

A 120-Million-Parameter Predictive Model of the Unitree Go2

Ramakrishna Kompella
Gödel Machines

Abstract

How do model size and training exposure affect a quadruped world model's ability to forecast motion from joint commands? We train five recurrent predictors of the Unitree Go2, spanning 8–120 million per-step parameters and four budgets of 0.36–2.88 billion sampled transitions. All 20 checkpoints forecast 30 proprioceptive channels on the same 24,576 held-out simulation windows, given the recorded commands and terrain scans. With size n in units of 30M parameters and exposure d in billions of transitions, three-second normalized error follows $0.202 + 0.00775n^{-0.863} + 0.229d^{-0.246}$ over this grid. Withholding the final exposure doubling gives a mean relative prediction discrepancy of 2.9%, compared with 7.4% for carrying forward the preceding scores. The final doubling reduces error at 120M from 0.401 to 0.364. Removing the scan yields a 119.7M-parameter model that can forecast from proprioception and commands alone. Updating its input projection improves one-second error on held-out flat ground from 0.831 to 0.613. At three seconds the corresponding errors are 0.948 and 1.009. We then replay the joint targets recorded on one physical Go2. Across nine stock-controller traces, the 8M and 120M adapted models achieve joint-angle errors of 0.057 and 0.064 rad, respectively, compared with 0.059 rad for MuJoCo with recorded gains. On the four 0.5 m/s trials the ordering changes to 0.051, 0.046 and 0.073 rad. The sweep quantifies scaling under recorded terrain information; the physical trials measure how sensor-only forecasts transfer under specific controller commands.

1 Introduction

When a quadruped bends its front legs, its subsequent pitch depends in part on whether those feet are bearing weight. A joint command and a single pose leave some of that information unresolved. A recent history of joint motion and body rotation can help a predictor distinguish standing, stepping, and recovering from a disturbance. Predicting what happens one control tick later is the simplest test of this idea. Predicting three seconds later is a different test: at 50 Hz the model must use its own outputs as inputs 150 times. A small error near foot contact can change every prediction that follows.

Two resources can improve such a predictor. Increasing its parameter count gives it more freedom to represent different responses to the same command. Increasing training exposure gives the optimizer more opportunities to learn those responses. Both cost compute. If the budget is fixed, increasing model width can mean seeing fewer examples. Empirical scaling laws describe how the resulting prediction error varies across these choices [13, 16, 22]. For a physical trajectory, even the unit of data requires care: a 20 ms transition may appear in many overlapping windows. We therefore report nominal sampled transition exposures separately from the number of distinct recorded interactions.

We measure this trade-off in a simulated Go2 with 12 actuated joints. The target is a time series with physical meaning: three components of body orientation relative to vertical, three angular rates, twelve joint positions, and twelve joint velocities. A shared evaluator makes it possible to compare models that differ by a factor of fifteen in size, even though the trajectories range from quiet postures to large changes of attitude. We ask how well a single error surface describes that grid, how its shape changes with forecast horizon, and which observations survive changes in the accounting of training exposure.

The terrain scan in the main study is sampled at the recorded pose on every future tick. It tells the predictor something about where the robot actually went. To use the forecaster on a physical Go2 with encoders and an IMU, we remove this input and adapt the first projection at each width. Comparing the original, zero-scan and adapted versions on the same simulated trajectories isolates the change in information available at inference. On the real robot we can ask a more direct question. Once the controller has issued its joint targets, how closely do the learned model and a physics simulator predict the motion that followed? The controller runs once on hardware. Each predictor receives those recorded targets, and the robot supplies the reference readings.

We first estimate the error surface and test a prediction at withheld training budgets. We then examine transfer over forecast time, across model sizes and across motion types. The physical evidence comprises nine default-mode SDK trials, two handset trials and two short named SDK requests on one robot. Eight additional gait requests have

a simulated reference. A filmed physical walk provides an independent measurement of travel alongside the encoder and IMU comparison.

2 Predicting a Go2 trajectory

2.1 Observations, commands and terrain

At time t, suppose a controller sends the robot twelve target joint angles. One control period later, the legs have moved and the body may have pitched, rolled or turned. Our prediction target is the resulting change in a vector of 30 readings, sampled at 50 Hz. With R_t the rotation from the body to the world frame and e_z the upward world unit vector, the readings are

$$x_t = \left(R_t^\top e_z,\ \omega_t,\ q_t,\ \dot{q}_t\right) \in \mathbb{R}^{30}. \tag{1}$$

The four blocks contain body orientation relative to vertical (3), body angular velocity (3), joint angles (12), and joint angular velocities (12). The action $a_t \in \mathbb{R}^{12}$ is the vector of joint position targets in radians for the transition from x_t to x_{t+1}. Each target drives a low-level position controller over ten 2 ms physics steps. All observations in this study are derived from simulator state. The sensor-only interface assumes access to a body attitude estimate, a gyroscope, joint encoders and the controller's own commands. The simulated sensor channels have no added measurement noise or delay. Noise injected into collection actions perturbs the motion itself.

The scaling experiment supplies an additional input $s_t \in \mathbb{R}^{187}$. A 17×11 grid, sampled every 0.1 m and aligned with the recorded body yaw, describes the local terrain. At grid point j the value is $\mathrm{clip}(z_{\mathrm{base}} - z_{\mathrm{terrain},j} - 0.5, -1, 1)$ with heights in metres. On level ground every entry therefore carries the same clipped base-height signal. During a forecast, the evaluator supplies future commands and scans from the recorded trajectory. The model must predict the evolution of x, including the effects of contact, conditional on both input sequences. Section 2.6 describes a separate model whose forecast uses only the command sequence after initialization.

2.2 Trajectory collection and sampling

We simulate the Unitree Go2 robot model [26] in MuJoCo Warp [7, 25]. The mixed training corpus comprises 1,634 shards and includes locomotion generated by a gait controller, perturbed controller actions, alternative continuations from states near terrain features, and random joint commands on flat ground. The last source includes fallen poses and recovery attempts. Randomized collection varies quantities such as ground friction, payload, motor gains and action delay. Earlier states and commands can thus carry information about conditions that are absent from an individual observation.

Within each shard, worlds whose index is divisible by 16 are reserved for validation. Branched continuations inherit their source world's split. Training and evaluation draw contiguous windows within single episodes. The loader splits trajectories at recorded contact overflows and rejects episode segments containing nonfinite states, base speeds above 5 m/s, angular speeds above 20 rad/s, or joint speeds above 40 rad/s. This filtering also limits the motion regime represented by the reported scores. A training cache contains 40 shards, apportioned by source, with a same-source replacement scheduled every 25 updates. Sampling with replacement permits overlapping windows and repeated visits to the same motion.

2.3 A recurrent predictor of changes

A one-second history helps distinguish superficially similar poses with different recent dynamics. A gated recurrent unit (GRU) [4] reads 50 preceding (x, a) pairs and initializes a 128-dimensional hidden state h_t. The initial reading x_t then anchors the forecast. At each step a residual network processes $(x_t, a_t, a_{t-1}, s_t, h_t)$. We standardize the observations and actions channel by channel using training statistics, and retain the scan's clipped physical encoding. The body first projects its input to width w, then applies seven residual blocks of the form $v \leftarrow v + \mathrm{GELU}(W\,\mathrm{LN}(v) + b)$, followed by layer normalization and a linear readout. Thirty outputs predict the standardized change in the readings. The remaining 256 outputs feed a GRU cell that updates h_t. We undo the change normalization, add the predicted change to x_t, and project the first three coordinates back onto the unit sphere.

Training gives equal weight to two losses on the same window. In the first pass, each step receives the recorded reading and incurs squared error on the standardized change $x_{t+1} - x_t$. In the second pass the network starts from

the same history and initial reading, then consumes its own predictions. It incurs a smooth L1 loss on each predicted reading after division by that channel's training standard deviation, with the quadratic to linear transition at unit error. Both losses are averaged over channels, time and windows. The second pass trains the model on the consequences of its earlier errors.

2.4 Parameter count, exposure and optimization

We vary w over 1,024, 1,440, 2,048, 2,880 and 4,096. Throughout the scaling analysis N counts parameters used at every prediction step, including the recurrent update. These counts are respectively 8.18, 15.64, 30.90, 60.16 and 120.37 million. Every model has a further 66,048 parameters in its history encoder. Each optimizer update samples b windows with H target transitions per window such that $bH = 131{,}072$. H grows from 16 to 32, 64 and 128 at 500-update intervals, then stays at 128. We count each target transition once even though both loss terms use it. The 50 context steps in each window enter the computation separately.

Each width has one training run, with seed zero. We use AdamW [20] with zero weight decay, gradient norm clipping at one, and bfloat16 autocasting in the residual body. During rollout backpropagation, we cap each window's observation and hidden-state gradient norms at ten times the respective batch median at each step. After 20 consecutive skipped updates, the optimizer and network can be restored to a previous clean snapshot. In ascending order of width, the peak learning rates are $(10, 7, 5, 3.5, 2.5) \times 10^{-4}$. Following a warmup–stable–decay schedule [10, 14], the rate rises linearly for 300 updates and stays at its peak on the shared training trajectory. At $u = 2{,}500$, $5{,}000$, $10{,}000$ and $20{,}000$ updates, a copy receives $0.1u$ more updates with a linear decay to zero. We evaluate this copy and continue the main run from the pre-decay state. The four checkpoints per width consequently share an initialization and most of their optimization history.

The nominal exposure is $D = 131{,}072\,(1.1u)$, giving 0.360448, 0.720896, 1.441792 and 2.883584 billion sampled target transitions. D measures a training budget with replay. It counts attempted updates, including steps later skipped for large gradients or superseded by a rollback. A branch checkpoint includes its own decay steps and excludes previous branches' decay steps. The learning curves compare exposure at a fixed corpus and also reflect the width-specific learning rates. The training log estimates compute per update as $6Nb(2H + 50)$, counting two prediction passes and one context pass. This proxy charges the full per-step network for the context pass, which in this architecture uses the smaller history encoder. Its relationship to executed FLOPs depends on H.

2.5 Held-out metrics

All 20 sweep checkpoints are scored on identical windows. We draw 96 shards uniformly without replacement (seed 2026), then draw 256 valid windows from held-out worlds per shard, for $M = 24{,}576$ windows. A common set of per-channel standard deviations from the training normalizer of a reference width-2,048, depth-4 checkpoint fixes the units for the entire family. Let $\hat{x}_{i,k}$ denote the recursively predicted reading k steps after the final observed reading in window i, and let $x_{i,k}$ denote its recorded counterpart. With observation scale $\sigma_{x,j}$, endpoint error at horizon k is

$$L_k = \left[\frac{1}{30M} \sum_{i=1}^{M} \sum_{j=1}^{30} \left(\frac{\hat{x}_{i,k,j} - x_{i,k,j}}{\sigma_{x,j}} \right)^2 \right]^{1/2}, \qquad k \in \{50, 150\}. \tag{2}$$

These endpoints correspond to one and three seconds. For one-step error we give the model the recorded readings at each of the 150 prediction times, carry its recurrent state, and compare its predicted change with the recorded change. We divide by the reference standard deviation of the *change* in each channel, then pool squared errors over all windows, times and channels before taking a square root. One-step and endpoint scores therefore use different units of normalization. Validation shards can contain multiple windows from the same simulated worlds. Bootstrap intervals for the individual sweep scores resample entire shards (2,000 draws) and quantify variation over this panel conditional on the trained weights.

2.6 Adapting the largest model to proprioception alone

We take the width-4,096 checkpoint at the largest exposure. Its input projection originally maps 369 coordinates (30 readings, 24 current and previous command coordinates, 187 scan entries and 128 hidden coordinates) to width 4,096.

We delete the 187 scan columns from its weight matrix, copy the bias, and retain all other weights and normalization buffers. Consequently the initial sensor-only prediction equals the source prediction with every scan value set to zero. This reduces the total model to 119,670,302 parameters. Its external interface now needs 30 readings and the current and previous joint commands at a prediction step.

For the 120M parent we update only this projection's $4{,}096 \times 182$ weights and 4,096 biases (749,568 trainable parameters). One flat-ground shard supplies one uninterrupted 200-transition segment from each of 64 training worlds and 32 held-out worlds. Within the training segments we sample 16 windows per update with 50 context steps and four prediction steps. We run 32 float32 AdamW updates at 10^{-4} with zero weight decay and the same two-part prediction objective. This amounts to 2,048 sampled target transitions. The comparison uses the same 32 held-out segments before and after adaptation, the source model's normalization, and two additional references. Persistence predicts $\hat{x}_{i,k} = x_{i,0}$ at every horizon. The original checkpoint receives each recorded scan. This flat-ground panel and the mixed-terrain scaling panel have different sampling distributions and reference scales.

3 How error scales with model size and training exposure

Consider two ways to spend an additional training budget. A larger network can represent a more complicated transition. More updates allow the network to learn from more examples of what follows a joint command. The two quantities can move independently in this experiment because each width supplies four checkpoints at increasing training exposure. Throughout this section, N counts parameters used by a recurrent prediction step and D counts nominal sampled target transitions, including repeat draws. All results in this section use scan-conditioned forecasts.

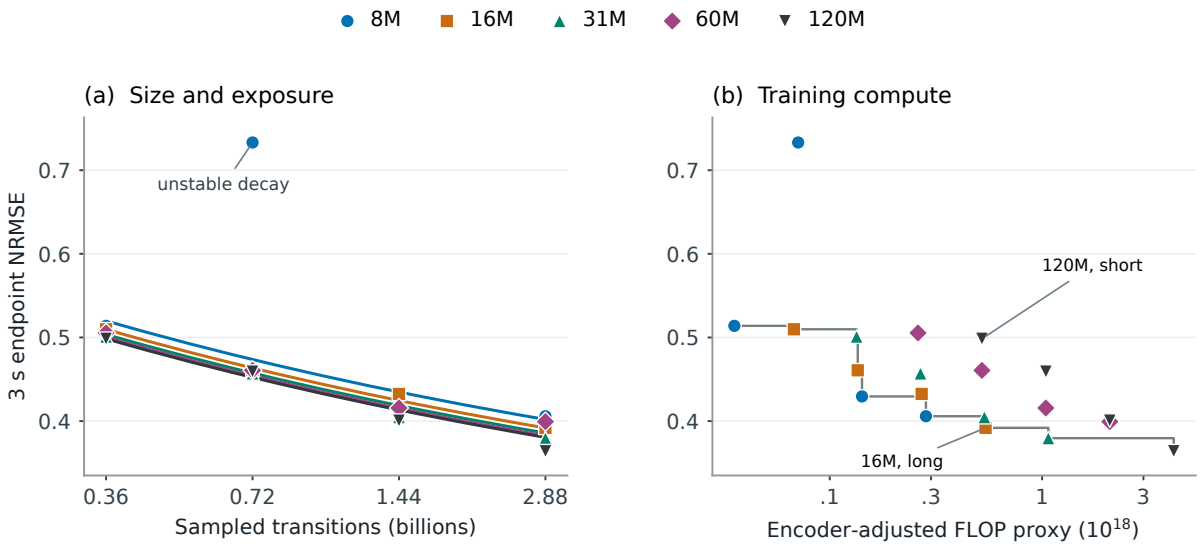


Figure 1: Three-second endpoint prediction error across all 20 checkpoints. Markers distinguish the five network sizes. (a) Each curve evaluates Eq. (3) at a fixed width. The high blue point is the 8M model's unstable decay branch, included in the fit. (b) The same checkpoints, located by estimated training compute. The step line gives the lowest error observed at or below each budget. The two labeled models have similar estimated compute and distinctly different prediction error.

Table 1: Fitted coefficients for Eq. (3) and absolute relative discrepancies, $|\widehat{L}/L - 1|$, across the 20 checkpoints used in the fit. Each row fits its own metric. One-step error scores changes in the observation with teacher forcing, while the rollout metrics score the predicted observation at a single endpoint.

	E_h	A_h	B_h	α_h	β_h	Median	Maximum
One step	.1442	.01254	.06811	.544	.541	0.67%	48.8%
1 second	.2816	.00701	.08778	.988	.502	0.77%	33.5%
3 seconds	.2020	.00775	.22874	.863	.246	1.19%	35.4%

3.1 An empirical formula

A simple model asks how much error remains after separately increasing capacity and exposure. For each evaluation metric h we fit an offset plus two decreasing terms, following the additive scaling form of Hoffmann et al. [13],

$$\widehat{L}_h(N, D) = E_h + A_h \left(\frac{N}{30 \times 10^6} \right)^{-\alpha_h} + B_h \left(\frac{D}{10^9} \right)^{-\beta_h}. \tag{3}$$

The reference units put A_h and B_h on the error scale. At 30M parameters and a billion exposures the fitted error is simply $E_h + A_h + B_h$. At a fixed size, doubling exposure multiplies the last term by $2^{-\beta_h}$. For the three-second fit this factor is 0.843. The total error falls by a smaller proportion because the other two terms stay fixed. E_h is the surface's extrapolated intercept at infinite size and exposure. The experiment does not measure this limit. The additive form also assumes that a fixed increase in exposure gives the same absolute error reduction at every width. Figure 1 shows the individual measurements against that assumption.

We fit the scores after pooling mean squared error over the 96 shards and taking the square root. Each checkpoint contributes one equally weighted residual $r = \log \widehat{L}_h - \log L_h$. We minimize the sum of Huber penalties [15], $r^2/2$ for $|r| \le 0.001$ and $0.001(|r| - 0.0005)$ otherwise. This criterion limits the influence of an individual failed training branch. The original fit uses bounded L-BFGS from 243 starting points, positive amplitudes and offset in log space, and exponents in $[0, 2]$. Table 1 expresses that fit in the units of Eq. (3). The unstable 8M branch at 0.72B exposures accounts for the maximum discrepancy in every row. At three seconds its observed error is 0.733, against a fitted value of 0.474.

3.2 Reading the surface at finite budgets

At the largest training budget, the three-second errors in order of increasing width were 0.406, 0.392, 0.380, 0.399, and 0.364. The 120M model attained the lowest error on all three metrics in the sweep. From 1.44B to 2.88B exposures, its one-step, one-second, and three-second errors fell from 0.205 to 0.183, 0.353 to 0.321, and 0.401 to 0.364, respectively. The three-second improvement was 9.1%, with lower error on 92 of the 96 paired shards. A 10,000-draw paired shard bootstrap gives a 95% percentile interval of $[0.018, 0.056]$ for the absolute decrease. Each draw samples the same 96 shard indices with replacement for both checkpoints and computes their difference in pooled RMSE. This interval describes variation across sampled evaluation shards, conditional on the trained checkpoints. Each width has a single training seed.

One use of the fitted surface is to interpret allocations at comparable compute. The 16M model with 2.88B exposures reached 0.392 three-second error, while the 120M model with 0.36B exposures reached 0.498. The relative reduction was 21.4%. The longer-trained model had lower squared error on all 96 shards (absolute RMSE decrease 0.107, paired 95% interval $[0.080, 0.132]$).

We estimate compute using the parameter-count convention of approximately six floating-point operations per parameter per processed training example (forward and backward combined). An update with b windows, horizon H, and context length K makes two prediction passes over bH targets, and one history-encoding pass over bK steps. Figure 1 therefore accumulates $6b(2HN + KN_{\text{enc}})$ over each checkpoint's training path, with $N_{\text{enc}} = 66{,}048$ for the context GRU. The estimates for the two compared models are 5.42×10^{17} and 5.21×10^{17} FLOPs, respectively. The original training log charged the cost of the larger per-step network to the context pass as well, giving 6.73×10^{17} and 8.26×10^{17}. The inferred ordering of compute cost is sensitive to that convention. Both estimates omit implementation costs such as activation recomputation, optimizer work, and data processing. Device time was not used for this comparison.

3.3 How precisely is the law identified?

To test extrapolation across the final exposure doubling, we withheld the five 2.88B checkpoints and refit the three-second surface to the 15 lower-budget observations. In ascending size order the predictions were 0.393, 0.384, 0.380, 0.379, and 0.378. Their mean absolute relative discrepancy from the withheld values was 2.9%, versus 7.4% for carrying forward the preceding observed error at each width. The largest relative discrepancy was 5.2%, for the 60M model. This retrospective check reuses the five training runs and the common evaluation panel. Independent training runs would test how well the rule guides a new allocation.

We also tested how strongly the three-second coefficients depend on numerical precision and training instability. For these refits we minimized the same robust log-residual loss in the centered units of Eq. (3), with seven initializations, an analytic Jacobian, amplitudes and offset bounded to $[10^{-6}, 5]$, and exponents bounded to $[0, 2]$. With all 20 points, this solver returned $(E, \alpha, \beta) = (0.178, 1.186, 0.222)$, with a 0.54% lower objective than the original fit. Removing the unstable 8M branch gave $(0.233, 0.499, 0.289)$. For a second sensitivity test we also replaced nominal exposure with the number of trunk updates surviving skips and rollbacks, plus scheduled decay updates, multiplied by 131,072. Decay skips were unlogged. This proxy counts training targets associated with retained updates. With this adjustment E reached its numerical lower bound, $\alpha = 0.051$, and $\beta = 0.253$. Omitting one width at a time from the nominal-budget fit gave α values from 0.035 to the upper bound of 2. Thus the measured variation over the grid constrains the finite-budget errors more strongly than it constrains the offset or the capacity exponent. We interpret the surface over the sampled range and leave predictions at substantially larger training scales open.

4 Forecasting from proprioception alone

On flat ground, the recorded terrain scan contains 187 copies of a clipped affine function of the robot's true base height. The scan therefore remains informative even on a surface with no geometric variation. A forecaster that receives only joint targets, joint encoders and a body attitude estimate must infer the consequences of vertical body motion from their history. The two interfaces can be compared on exactly the same flat-ground trajectories.

Table 2: **Removing the recorded terrain scan.** Endpoint errors use the same 32 held-out flat-ground worlds, targets, and checkpoint normalization for all four rows. The first row repeats the reading at the forecast origin. The final row uses the original model and its recorded scan throughout. These trajectories are a separate evaluation panel from the scaling study.

Forecast	1 s NRMSE	3 s NRMSE
Last observed reading (persistence)	1.170	1.253
Proprioception, before adaptation	0.831	0.948
Proprioception, after 32 updates	0.613	1.009
Original model with recorded scan	0.131	0.136

At one second, adaptation reduces normalized RMSE by 0.219 relative to direct transfer (Table 2). A paired percentile bootstrap over the 32 held-out worlds gives a 95% interval of $[-0.402, -0.027]$ for adapted minus initial error. At three seconds, the observed difference reverses sign ($+0.061$), and the paired interval is $[-0.322, 0.439]$. Relative to persistence, the adapted model has lower point estimates at both horizons. The corresponding intervals are $[-0.711, -0.395]$ at one second and $[-0.502, 0.059]$ at three seconds. We draw 10,000 samples of worlds with replacement, retain each world's squared error for all models, and take the square root after averaging within each resample. These intervals quantify variation across worlds in this one shard, conditional on the trained weights.

The one-step normalized delta error fell from 0.426 to 0.278. The adaptation loss covers four forecast ticks (80 ms), whereas a three-second rollout consumes 150 successive predictions. Figure 2 shows the resulting time dependence. Throughout rollout, the original predictor has access to the recorded scan sequence, which continues to convey information about the realized body height.

Does the same update help every size? We apply the identical projection-only update to the longest-trained checkpoint at each width. One-step error decreases at every size (Table 3). At three seconds, it decreases at 8M and 60M

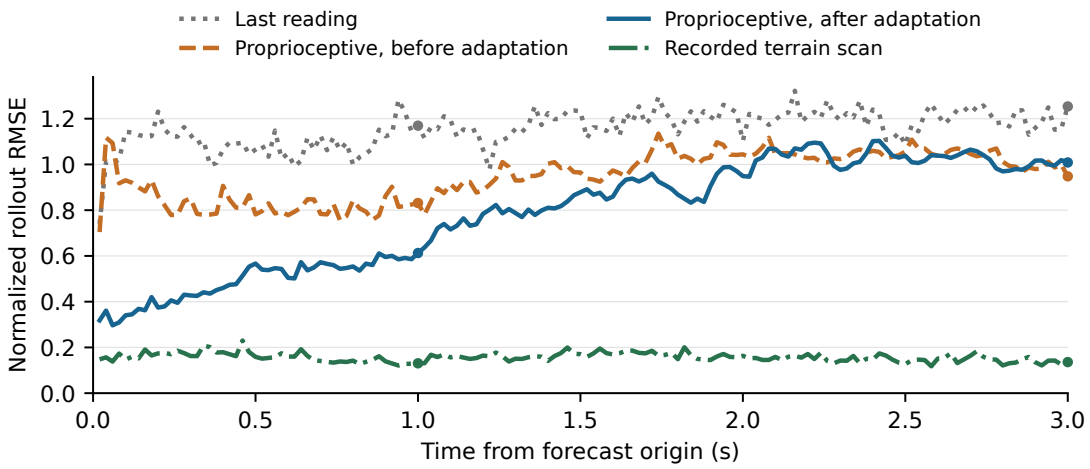


Figure 2: **Forecast error through time on the flat-ground panel.** Each curve aggregates the channel-wise normalized squared errors across all 32 trajectories before taking the square root. Dots mark the endpoints in Table 2. Differences between adjacent points reflect the particular states reached at each instant as well as accumulated prediction error.

and increases at the other three widths. A lower one-step error following adaptation is therefore insufficient to infer a lower three-second error. The hardware comparisons below use these same five adapted checkpoints.

Table 3: Sensor-only validation error before and after 32 projection updates. Each pair uses the same parent checkpoint, initialization data and validation segments. The one-step column scores a normalized change in the readings. The other columns score the normalized readings at the stated endpoint.

	One step		1 s		3 s	
Parent	Before	After	Before	After	Before	After
8M	0.611	0.377	0.833	0.758	1.011	0.725
16M	0.477	0.334	0.854	0.974	1.035	1.254
31M	0.447	0.277	0.789	0.688	0.901	1.168
60M	0.514	0.354	1.897	1.544	1.917	1.310
120M	0.426	0.278	0.831	0.613	0.948	1.009

5 An open-loop test on a physical Go2

A physical trial supplies both an input and an answer. The controller publishes joint targets as the Go2 moves, and the encoders and IMU record the robot's response. We replay that target sequence through each predictor. This removes differences in command generation from the comparison. Each forecast remains conditional on the commands chosen during the real trajectory, including the controller's responses to earlier disturbances.

Inputs and reference. The primary three-second panel uses nine aligned default-mode SDK recordings on a single Go2 through the filmed 2 m trial. Two subsequent named-mode trials provide shorter forecasts (Table 7). The supplement also provides two separately identified handset recordings. The panel includes slow and stalled trials and an operator-interrupted walk. An earlier recording lacks controller identification and is excluded. We take one window per trial, starting at the walk cue. A second of joint encoders, IMU-derived up direction, body gyro and published targets initializes the recurrent model. The learned models and both physics replays receive the same 150 joint-target vectors at 50 Hz. We compare their predictions with the next 150 robot readings, with no state corrections.

Streams are sampled on the host's monotonic clock using the last reading at or before each tick, with a maximum age of 40 ms. Publication on the command topic does not independently establish actuator acceptance.

MuJoCo starts at the measured joint positions, joint velocities, body attitude and angular velocity. We set initial base linear velocity to zero and place the lowest foot on the floor. In the three-second panel we compare its nominal actuator gains ($K_p = 25$, $K_d = 0.5$) with the constant published gains $(40, 1)$; all published velocity targets and feedforward torques are zero. Both replays use the same motor limits and ten 2 ms physics steps per target. The learned predictor outputs joint angles, joint velocities, gyro and body-up direction. MuJoCo integrates a floating base and joints, from which we derive the corresponding sensors. Only the simulator represents global base translation. Real travel below comes from odometry or video.

5.1 Three seconds on the default controller

Table 4 gives errors pooled over nine SDK trials, alongside joint-angle scores for four SDK trials at 0.5 m/s and two separately collected handset traces. Among the five adapted sizes, the smallest has the lowest pooled joint-angle error on the nine SDK trials. It beats the largest on four of nine individual traces, with larger improvements on those four (Fig. 3). The 16M model has the lowest gyro and up errors among the adapted models, and 120M has the lowest joint-velocity error. Persistence scores well on the initially upright, slowly rotating body while missing most leg motion.

Table 4: Open-loop predictions in physical units over three seconds. The four central columns pool nine stock-SDK recordings. The last two columns give joint-angle RMSE on four of those nine trials commanded at 0.5 m/s and on two additional handset recordings. RMSE pools squared error across ticks, channels and trials. Up error is the mean frame-wise angle between the predicted and measured up vectors.

	All nine SDK trials				Joint angle (rad)	
Predictor	q (rad)	$\dot{q}$ (rad/s)	Gyro (rad/s)	Up (deg)	SDK 0.5 m/s	Handset
Persistence	0.1539	1.193	0.131	2.02	0.1540	0.1295
MuJoCo, nominal	0.1220	1.134	0.814	31.86	0.1461	0.1222
MuJoCo, recorded gains	0.0586	0.733	0.557	5.16	0.0729	0.0429
8M	0.0566	0.773	0.243	4.41	0.0512	0.0591
16M	0.0653	0.576	0.192	2.12	0.0529	0.0667
31M	0.1412	1.135	0.491	14.84	0.1485	0.1585
60M	0.1278	0.856	0.318	7.56	0.0945	0.0626
120M	0.0641	0.569	0.206	2.24	0.0461	0.0545

The four 0.5 m/s trials put the largest adapted model first on joint angles. On the two handset traces, recorded-gain MuJoCo achieves 0.0429 rad and the 120M model 0.0545 rad. On the nine-trial panel, substituting the published gains more than halves MuJoCo's joint-angle error, from 0.1220 to 0.0586 rad. This change in the physics baseline is larger than the 0.0075 rad gap between the smallest and largest adapted models on the same recordings. Appendix B gives the corresponding comparisons for the 20 zero-scan parents, three archival sequence models, two sensor models and four scratch pilots.

5.2 A filmed two-metre walk

Three stock SDK trials reached their 2 m odometry target in 5.00–5.08 s at a requested 0.5 m/s. A fourth ended on operator interrupt at 1.67 m. Table 5 gives both stop and settled odometry. The agreement among the three completed trials measures the repeatability of the odometry-triggered stop. One trial also has the independent grid measurement in Fig. 4.

The four-contact centroid moved an estimated 2.335 m in the plane. Independent picks and a new perspective fit gave 2.344 m. Shifting the picks coherently by two pixels at each endpoint gave 2.286–2.397 m. This sensitivity range reflects endpoint selection. Tape scale, perspective residuals, lens distortion and stance changes also affect the comparison. Settled base odometry was 2.035 m, about 0.30 m below the camera estimate on this one trial. The robot was still stepping at the stop frame.

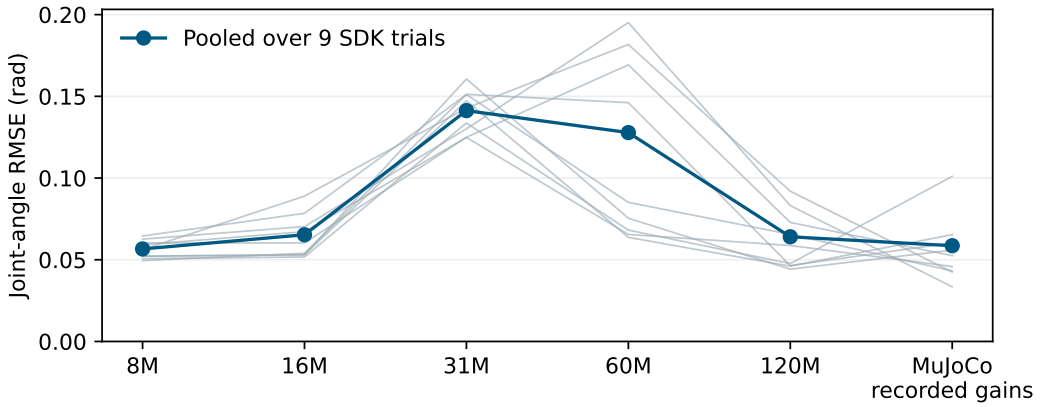
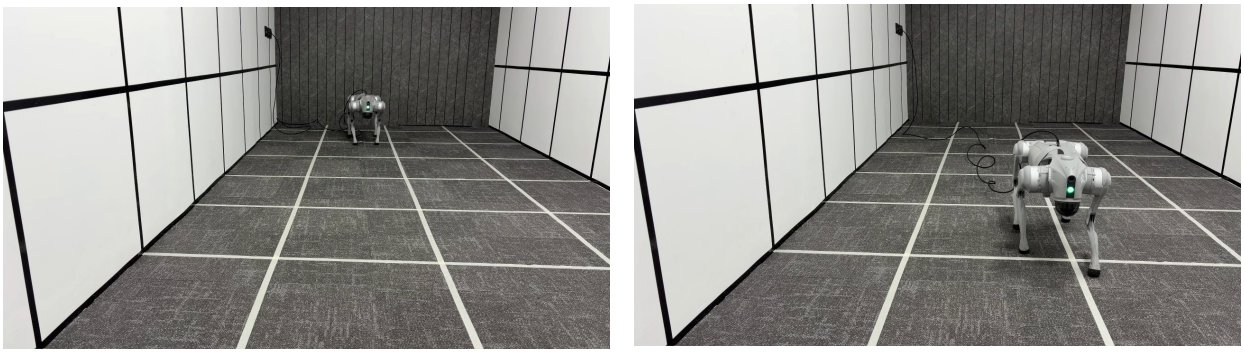


Figure 3: Each gray line follows the same physical command sequence across the five adapted models and the physics replay with recorded gains. The heavy line pools squared joint-angle errors before taking the square root.



Before the start

After settling

Figure 4: Stationary endpoints of the filmed 0.5 m/s SDK walk. Each tape interval measures a reported 2 ft (0.6096 m). We project each of the four foot contacts onto the floor with a separate homography for each frame, then measure displacement between their centroids. This construction avoids projecting the elevated trunk onto the floor.

The stop-window odometry speed was 0.394 m/s. Tracking the front indicator to identify visible motion and dividing the stationary-endpoint travel by that interval gave about 0.47 m/s (0.44–0.48 m/s over 2–4 pixel motion thresholds). The clocks place first visible motion 0.284 s after the start command. That interval includes unknown phone transport delay and clock alignment uncertainty (47 ms SSH half round trip). Odometry entered and stayed within 1 cm of its final value by 0.12 s after the stop. The settled video frame was taken 2.50 s after the stop, beyond the last telemetry sample at 2.13 s, by which time odometry was stationary.

Table 5: Four physical SDK walks at a requested 0.5 m/s. Stop and settled travel are planar onboard odometry. Only the last walk has the independent endpoint measurement used in Fig. 4.

Trial start	Stop condition	Command (s)	At stop (m)	Settled (m)
17:52:37	Operator interrupt	4.329	1.666	1.697
17:53:21	2 m target	5.019	2.001	2.043
18:02:47	2 m target	4.996	2.001	2.042
18:06:25	2 m target	5.081	2.002	2.035

The same filmed trial also provides 254 forecast ticks of sensor data (Table 6, Fig. 6). Recorded-gain MuJoCo and the 120M model give nearly identical mean up errors, within $0.05°$. Nominal-gain MuJoCo diverges in attitude even though the physical robot's peak tilt is only $2.71°$. Across all four 0.5 m/s trials, the 120M model's first-three-second

joint-angle errors span 0.044–0.048 rad. Its mean up error exceeds that of recorded-gain MuJoCo on two of the four. Global travel is available from the camera and odometry. The learned model forecasts the sensor channels in the table.

Table 6: Physical sensor prediction over the entire 5.08 s filmed walk. Each predictor receives the same published joint angles from the physical SDK controller.

Predictor	q (rad)	$\dot{q}$ (rad/s)	Gyro (rad/s)	Up (deg)
120M adapted	0.0444	0.754	0.259	2.70
MuJoCo nominal	0.1381	1.611	0.946	125.09
MuJoCo recorded gains	0.0628	0.916	0.391	2.75

5.3 Two named controller requests on hardware

The SDK acknowledged one `StaticWalk` and one `TrotRun` request on the same flat lane. Both trials requested 0.5 m/s forward and a 0.45 m goal. StaticWalk reached the distance stop in 1.31 s (0.459 m planar displacement in the controller log). TrotRun reached the 2.01 s time limit at 0.362 m. We score the first 50 forecast ticks from each cue, including the transition into the requested mode. Both sport-state gait codes remained zero; the mode labels identify the acknowledged requests. One trajectory per request gives a paired predictor comparison within each trajectory and leaves between-trial variability unmeasured.

The published joint commands in these trials carry time-varying gains, target velocities and feedforward torques. As a second physics baseline we replay all these channels at 50 Hz, using $\tau = K_p(q^* - q) + K_d(\dot{q}^* - \dot{q}) + \tau_{\text{ff}}$ before the same motor limits. This gives the physics baseline more command information than the learned models receive. On the common first second, 8M has the smallest joint-angle error among the five learned sizes for both requests. The horizon matters: over the full 2.00 s scored for TrotRun, the 120M and nominal MuJoCo joint-angle errors are 0.193 and 0.186 rad, respectively. Persistence gives the smallest gyro error on both first-second windows, which include the initially stationary robot.

Table 7: First second of each named SDK request on the physical Go2. Each has one recording. The learned models receive position targets. The full-drive physics replay also receives target velocities, gains and feedforward torques. Appendix C reports the full available horizons.

	StaticWalk request				TrotRun request			
Predictor	q	$\dot{q}$	Gyro	Up	q	$\dot{q}$	Gyro	Up
Persistence	0.1898	2.428	0.155	1.70	0.2220	3.735	0.237	1.98
MuJoCo, nominal	0.2111	2.791	0.928	12.56	0.1998	4.288	1.060	12.99
MuJoCo, full drive	0.2323	1.580	0.761	9.76	0.2156	2.240	1.465	12.46
8M	0.1212	2.374	0.555	2.66	0.1392	4.258	0.522	1.82
16M	0.1481	2.380	0.556	3.17	0.1500	3.986	0.487	2.59
31M	0.1574	2.684	0.652	9.01	0.1720	4.532	0.698	9.40
60M	0.1760	2.997	0.997	3.74	0.1721	4.820	0.775	3.85
120M	0.1279	2.286	0.555	3.63	0.1773	4.255	0.555	5.57

Joint angle in rad, joint velocity and gyro in rad/s, up error in degrees.

6 Gait-dependent fidelity in simulation

The named SDK trials give two short physical examples. The T2 policy in MuJoCo allows a controlled comparison across eight motion requests. For each request we use the same standing initialization, a one-second history, a forward command of 0.5 m/s and a three-second forecast on flat ground. T2 generates the joint targets while coupled to MuJoCo. We replay those targets through the adapted 120M model and score its predictions against the simulated

trajectory. Each row of Table 8 therefore compares a learned forecast with the physics simulation for one policy-generated sequence.

Table 8: Three-second simulated gait replays, adapted 120M model against MuJoCo. All errors aggregate the full 150-tick window. The stair-climb policy request also runs on the flat lane.

T2 request	Joint angle (rad)	Joint velocity (rad/s)	Gyro (rad/s)	Up (deg)
Walk	0.0332	0.501	0.158	1.02
Trot	0.0268	0.574	0.102	0.52
Flying trot	0.0241	0.585	0.104	0.80
Pace	0.0335	0.801	0.272	2.83
Bound	0.0847	0.745	0.324	5.50
Pronk	0.1555	1.523	1.310	69.52
Crouch walk	0.0523	0.876	0.204	4.97
Stair climb	0.0295	0.530	0.107	0.77

Flying trot has the smallest joint-angle error. Trot has the smallest gyro and up errors. Pace illustrates the difference between accurate leg angles and accurate attitude: its joint-angle error is close to walk, while its up error is nearly three times larger. The largest discrepancy occurs in pronk, where the MuJoCo reference overturns. Its 69.5° mean up error records a missed attitude trajectory. Appendix C shows how the discrepancies develop over time. Each request contributes one trajectory. Physical coverage comprises the default SDK, handset and two named SDK requests. Matching physical T2-policy traces, independently identified footfall classes and repeated named-mode trials remain open.

7 Related work

Empirical scaling laws. Language-model scaling studies established two distinct uses of pretraining experiments. Brown et al. evaluated how task performance changes as a predictor grows [3]. Kaplan et al. measured regularities in held-out prediction loss across budgets [16], and Hoffmann et al. used smaller runs to choose how to divide a larger compute budget between parameters and training tokens [13]. The additive form of our fit follows the latter work. Here the data coordinate counts sampled transitions, including repeated draws from a finite collection. Its exponent therefore describes learning with additional exposure under this sampling procedure. Distinct-example count and repeat count can have different effects even at equal compute [21].

Pearce et al. established scaling curves for pretrained agents and world models and showed that the fitted coefficients depend on the task, representation, and architecture [22]. RoboJEPA fitted a compute-scaling relationship for visual latent prediction across robotic embodiments, tested it at larger scales, and related forecasting error to planning success [29]. The closest comparison in quadruped dynamics is the study of neural Koopman operators by Abuduweili et al., who varied training-sample count and latent dimension and measured prediction and control on a simulated Go2 [1]. Our axes are per-step network size and transition exposure, and our dependent variable is the error in predicted body and joint readings at a specified time after the forecast begins.

Action-conditioned robot models. Recurrent world models made it possible to train a controller on imagined trajectories [8]. DayDreamer demonstrated learning from physical quadruped interaction [27], and TD-MPC2 studied model and data size in multi-task continuous control [11]. Rapid Motor Adaptation used recent proprioceptive history to adapt a locomotion policy to changing physical conditions [17]. For legged robots, the Robotic World Model used learned dynamics for policy optimization [19], and QWM conditioned dynamics on morphology to support policy transfer across quadrupeds [5]. These papers evaluate what a controller can accomplish using a learned model. Our experiments measure the accuracy of the underlying forecasts under recorded actions.

Several studies use the Go2 itself. MoSim predicts future physical states from state–action sequences and includes a simulated Go2 benchmark [12]. SimDist uses joint-position targets on a physical Go2 and adapts a pretrained world model for locomotion on slippery and compliant surfaces [18]. Amigo et al. trained a billion-parameter visual world model for a Go2 pushing task, with body-velocity commands issued at 5 Hz to an existing walking controller [2]. Our model operates at the joint-command interface at 50 Hz. The distinction determines what a proposed action means

and what a forecast can be scored against. We evaluate a family given recorded terrain scans, then evaluate all five adapted sizes from proprioception and joint commands on the same recorded physical trajectories.

Comparing a predictor with physical motion. The commanded quantity determines what can be tested on hardware. Xu et al. [28] use terrain and motion history to forecast displacement and failure risk under candidate commands. Our forecaster predicts the response of individual joints and the body IMU. At this interface the simulated actuators matter: Tan et al. [24] modeled actuator behavior and latency for quadruped policy transfer, and Hackett and Hubicki [9] used contact-free Go2 motions to identify joint inertia and friction parameters. We supply the gains published during each walk and measure their effect on sensor prediction. This leaves other simulator parameters fixed. Dao and Fern [6] instead compare distributions of proprioception and actions across simulated and physical rollouts, which permits evaluation without time alignment. Our recorded-command protocol asks a complementary question about the sensor error at a particular time after an observed command.

8 Discussion

Reading a scaling curve as a training decision. Within the tested scan-conditioned family, the lowest measured error occurs at the largest width and the longest budget. A different question arises when the training budget is already committed. The measured grid then favors a smaller, longer-trained model in at least one pair with comparable estimated cost. The large model was still improving at the edge of the grid. This trade-off can be read directly from Eq. (3). To first order, compute grows with ND. On a curve of constant ND, an increase in $\log N$ entails an equal decrease in $\log D$. At an interior minimum, the marginal reduction in error from width equals the marginal cost of seeing fewer examples, giving $\alpha A n^{-\alpha} = \beta B d^{-\beta}$ in the reference units of the fit. The uncertainty in the capacity exponent makes the location of this minimum particularly sensitive to the fitted parameters. The observed frontier is the more direct guide for the budgets tested.

A prospective test would fit the surface on smaller runs, fix a larger compute budget, record the predicted winner and its error, then train that configuration and a competing allocation at the same budget. For the proprioceptive model, this requires a scaling series with that input interface. A retrospective fit on models supplied with terrain scans cannot specify the sensor-only exponents. Nor does a falling curve in sampled exposures by itself measure the value of collecting fresh data. To estimate that value, one could vary the number of distinct trajectories while holding the update count, terrain mixture, and evaluation windows fixed.

What the evaluation measures. Recorded commands make every model answer the same forecasting question. A feedback controller, however, chooses its next command from the state it observes. When such a controller is coupled to a learned model, even a small error can change the subsequent commands and move the trajectory into a different region of state space. An independent audit on simulated ANYmal D found that improving short-horizon predictions could still yield failed walking policies [23]. A direct test would freeze the predictor, use it to select controllers for a previously unseen objective, and compare their predicted ordering with independent physical trials. A calibrated physics simulator evaluated with the same real-data allowance would provide a practical reference. The hardware replay measures open-loop prediction on recorded controls. Closed-loop policy selection remains untested.

Scope. The main estimates use one training seed per width, one embodiment, and idealized simulated observations for the scaling and adaptation experiments. Windows share simulated conditions, and decay branches reuse optimization history. Uncertainty from resampling evaluation data therefore has a narrower interpretation than uncertainty about retraining the family. The sensor-only experiment further conditions on one shard of flat-ground data. A real instrument introduces bias, noise, delay and differences in joint-state and attitude estimates. A terrain-relative sensor cue built with a real device would also have different information and timing from the recorded scans supplied in the main experiment.

9 Conclusion

For scan-conditioned Go2 forecasts, a single surface describes the effect of model size and training exposure well enough to predict the measured final doubling. At similar estimated compute a longer-trained 16M model wins over a

briefly trained 120M model. With both size and exposure at their largest tested values, the 120M model achieves the lowest simulation error. Removing the terrain scan changes the prediction problem. A small input-projection update recovers part of the lost accuracy at one second, and the three-second outcome depends on model size. Replaying the robot's own commands shows where this distinction matters in physical units. Models at both ends of the size range predict joint motion competitively with a physics simulator supplied with the recorded gains. Their ordering changes with the motion, the scoring horizon and the sensor channel. A prospective compute allocation and repeated, independently identified physical gaits would test how far these findings extend.

References

[1] Abulikemu Abuduweili, Yuyang Pang, Feihan Li, and Changliu Liu. Scaling law of neural Koopman operators. *arXiv:2602.19943*, 2026. URL https://arxiv.org/abs/2602.19943.

[2] Joseph Amigo, Rooholla Khorrambakht, Nicolas Mansard, and Ludovic Righetti. Coupled local and global world models for efficient first order RL. In *10th Conference on Robot Learning*, 2026. URL https://arxiv.org/abs/2602.06219.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2005.14165.

[4] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

[5] Mohamad H. Danesh, Chenhao Li, Amin Abyaneh, Anas Houssaini, Kirsty Ellis, Glen Berseth, Marco Hutter, and Hsiu-Chin Lin. Morphology-conditioned world model for cross-embodiment quadrupedal locomotion. *arXiv:2604.08780*, 2026. URL https://arxiv.org/abs/2604.08780.

[6] Jeremy Dao and Alan Fern. Simulator adaptation for sim-to-real learning of legged locomotion via proprioceptive distribution matching. *arXiv:2604.11090*, 2026. URL https://arxiv.org/abs/2604.11090.

[7] Google DeepMind and NVIDIA. MuJoCo Warp. Software repository, 2025. URL https://github.com/google-deepmind/mujoco_warp.

[8] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

[9] Jacob Hackett and Christian Hubicki. What good is a good robot model? differentiable regression, parameter ablation, and policy performance of quadrupedal models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2026. URL https://www.optimalroboticslab.com/publications.

[10] Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.18392.

[11] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.16828.

[12] Chenjie Hao, Weyl Lu, Yifan Xu, and Yubei Chen. Neural motion simulator pushing the limit of world models in reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27608–27617, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Hao_Neural_Motion_Simulator_Pushing_the_Limit_of_World_Models_in_CVPR_2025_paper.html.

[13] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2203.15556.

[14] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06395.

[15] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

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

[17] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems XVII*, 2021. URL https://arxiv.org/abs/2107.04034.

[18] Jacob Levy, Tyler Westenbroek, Kevin Huang, Fernando Palafox, Patrick Yin, Shayegan Omidshafiei, Dong-Ki Kim, Abhishek Gupta, and David Fridovich-Keil. Simulation distillation: Pretraining world models in simulation for rapid real-world adaptation. In *Robotics: Science and Systems XXII*, 2026. URL https://roboticsproceedings.org/rss22/p017.html.

[19] Chenhao Li, Andreas Krause, and Marco Hutter. Robotic world model: A neural network simulator for robust policy optimization in robotics. *arXiv:2501.10100*, 2025. URL https://arxiv.org/abs/2501.10100.

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[21] Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A. Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.16264.

[22] Tim Pearce, Tabish Rashid, David Bignell, Raluca Georgescu, Sam Devlin, and Katja Hofmann. Scaling laws for pre-training agents and world models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48542–48562, 2025. URL https://proceedings.mlr.press/v267/pearce25a.html.

[23] Joyjeet Singh. Auditing a learned simulator for legged robots: The robotic world model on ANYmal D. *arXiv:2611.23118*, 2026. URL https://arxiv.org/abs/2611.23118.

[24] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems XIV*, 2018. URL https://www.roboticsproceedings.org/rss14/p10.html.

[25] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.

[26] Unitree Robotics. Unitree MuJoCo robot models and simulation interface. Software repository, 2026. URL https://github.com/unitreerobotics/unitree_mujoco. Accessed November 2026.

[27] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. DayDreamer: World models for physical robot learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240, 2023. URL https://proceedings.mlr.press/v205/wu23c.html.

[28] Yiming Xu, Zhuoxun Yang, Songhai Zhu, Haochuan Zhou, Dianhao Zhang, and Qiang Liu. Causally inspired feature calibration for out-of-distribution state-transition prediction in quadrupedal locomotion. *Applied Sciences*, 16(21):11012, 2026. doi: 10.3390/app162111012. URL https://doi.org/10.3390/app162111012.

[29] Artem Zholus, Nicolas Beltran-Velez, Jianhao Yuan, Sarath Chandar, Tushar Nagarajan, Daniel Severo, Koustuv Sinha, Michal Drozdzal, Adriana Romero Soriano, Jeannette Bohg, Nicolas Ballas, and Mahmoud Assran. RoboJEPA: Scaling robotic latent world models. *arXiv:2610.10515*, 2026. URL https://arxiv.org/abs/2610.10515.

A Measured scaling grid and scratch pilots

The fitted surfaces in Section 3 use the checkpoint errors below. Comparing columns within a row shows how the scoring horizon changes the error. Comparing rows at a fixed size shows what an additional exposure budget bought on the same evaluation windows.

Table 9: Observed normalized errors for the complete scan-conditioned grid. Each checkpoint was evaluated on the same 96 held-out shards. The 8.18M, 0.721B branch experienced an unstable learning-rate decay.

Per-step parameters (M)	Sampled transitions (B)	One-step delta	1 s endpoint	3 s endpoint
8.18	0.360	0.284	0.449	0.514
8.18	0.721	0.490	0.617	0.733
8.18	1.442	0.225	0.372	0.430
8.18	2.884	0.209	0.359	0.406
15.64	0.360	0.280	0.438	0.510
15.64	0.721	0.243	0.399	0.461
15.64	1.442	0.223	0.368	0.433
15.64	2.884	0.201	0.347	0.392
30.90	0.360	0.275	0.438	0.501
30.90	0.721	0.235	0.390	0.457
30.90	1.442	0.210	0.365	0.405
30.90	2.884	0.193	0.340	0.380
60.16	0.360	0.271	0.431	0.505
60.16	0.721	0.242	0.399	0.461
60.16	1.442	0.212	0.355	0.416
60.16	2.884	0.205	0.349	0.399
120.37	0.360	0.268	0.437	0.498
120.37	0.721	0.231	0.389	0.459
120.37	1.442	0.205	0.353	0.401
120.37	2.884	0.183	0.321	0.364

Four small pilots trained sensor-only GRUs from scratch (width 128, depth 3). Each used 600 updates, 16 context ticks and an eight-step training horizon. With 64 training worlds both seeds improved their three-second endpoint score relative to training on eight worlds. The pilot evaluations use 64 validation worlds and their experiment-specific normalization. The projection comparisons in Section 4 use 32 validation worlds and their parent normalization.

Table 10: Scratch pilot validation errors after 600 updates. The two seeds also index independently sampled cohorts. Supplementary training records include intermediate updates.

Seed	Training worlds	One-step delta	1 s endpoint	3 s endpoint
0	8	0.666	0.778	0.853
0	64	0.391	0.585	0.605
1	8	1.183	6.596	14.570
1	64	0.956	1.310	1.346

B All replayed model families

The fixed panel includes 34 learned configurations. Five are projection-adapted, 20 are zero-scan versions of the scaling checkpoints, and nine are earlier sequence models or small pilots. Persistence and two physics configurations bring the count to 37 predictors on each physical trace. In the eight T2 traces MuJoCo supplies the reference, giving 35 predictors per gait. Table 4 reports the five adapted sizes and the three physical baselines. Table 11 completes that inventory on the same recordings.

Table 11: Remaining 29 learned configurations. The first 20 rows are zero-scan parents, labeled by per-step size and billions of nominal sampled transitions. The other nine comprise three archived GRU checkpoints, two sensor GRUs and four scratch pilots. Each row uses the pooling and physical units of Table 4.

	All nine SDK trials				Joint angle (rad)	
Predictor	q (rad)	$\dot{q}$ (rad/s)	Gyro (rad/s)	Up (deg)	SDK 0.5 m/s	Handset
8M, 0.360B	0.0632	0.777	0.263	4.28	0.0510	0.0558
8M, 0.721B	0.1377	1.410	1.548	20.46	0.1488	0.1333
8M, 1.442B	0.0754	0.858	0.346	4.04	0.0774	0.0758
8M, 2.884B	0.0901	0.994	0.364	5.27	0.0848	0.0958
16M, 0.360B	0.0789	0.776	0.258	4.34	0.0574	0.0621
16M, 0.721B	0.0870	0.812	0.303	6.01	0.0827	0.0925
16M, 1.442B	0.0796	0.886	0.273	5.33	0.0797	0.0751
16M, 2.884B	0.0796	0.863	0.345	4.64	0.0748	0.0893
31M, 0.360B	0.0794	0.730	0.222	4.01	0.0562	0.0652
31M, 0.721B	0.0749	0.836	0.285	2.84	0.0735	0.0673
31M, 1.442B	0.0769	0.889	0.288	4.25	0.0827	0.0710
31M, 2.884B	0.0730	0.883	0.265	3.58	0.0722	0.0737
60M, 0.360B	0.0711	0.803	0.219	2.86	0.0675	0.0820
60M, 0.721B	0.0925	0.879	0.347	5.81	0.0654	0.0968
60M, 1.442B	0.0677	0.897	0.333	2.95	0.0587	0.0792
60M, 2.884B	0.1055	0.966	0.389	6.88	0.1007	0.0965
120M, 0.360B	0.0926	0.844	0.238	3.85	0.0772	0.0918
120M, 0.721B	0.0841	0.887	0.337	3.53	0.0824	0.0670
120M, 1.442B	0.0698	0.792	0.271	3.85	0.0640	0.0687
120M, 2.884B	0.0777	0.782	0.276	3.00	0.0722	0.0599
13M GRU, cosine 8k	0.0847	0.915	0.296	3.67	0.0895	0.0958
120M GRU, WSD 5k	0.0868	0.872	0.314	4.83	0.0900	0.0795
120M GRU, WSD 10k	0.0927	0.864	0.323	3.29	0.0909	0.0822
Sensor GRU, seed 0	0.0945	0.991	0.346	4.74	0.0951	0.0850
Sensor GRU, seed 1	0.0966	0.935	0.349	4.10	0.0827	0.0861
Pilot, seed 0, 8 worlds	0.1436	1.086	0.229	5.54	0.1203	0.1319
Pilot, seed 0, 64 worlds	0.1289	0.827	0.276	2.96	0.1255	0.1171
Pilot, seed 1, 8 worlds	0.1211	1.099	0.310	2.16	0.1093	0.1192
Pilot, seed 1, 64 worlds	0.1307	1.021	0.307	4.95	0.1202	0.1065

The column for all nine SDK trials also permits a paired comparison of the longest-trained parent with its adapted counterpart at each width. Joint-angle error decreases after adaptation at 8M, 16M and 120M and increases at 31M and 60M. At 31M it rises from 0.0730 to 0.1412 rad. A common adaptation recipe can change the ordering of the parent checkpoints.

C Full named-mode windows and time-resolved gait errors

Table 7 fixes the scoring window at one second for both physical recordings. Table 12 instead uses every complete 20 ms tick during each command. The TrotRun comparison then extends from the initial transition through two seconds of motion. The simulator in Fig. 5 supplies a separate three-second trajectory for each T2 request.

Table 12: Real Go2, full named-mode windows. Joint angle is in rad, joint velocity and gyro in rad/s, and body-up error in degrees. The 1.30 s and 2.00 s windows correspond to 65 and 100 scored ticks.

	StaticWalk, 1.30 s				TrotRun, 2.00 s			
Predictor	q	$\dot{q}$	Gyro	Up	q	$\dot{q}$	Gyro	Up
Persistence	0.1875	2.438	0.150	1.70	0.2240	3.803	0.233	2.58
MuJoCo, nominal	0.2188	2.776	0.938	13.55	0.1863	4.291	1.851	63.56
MuJoCo, full drive	0.2760	1.816	1.270	18.14	0.3101	2.326	1.868	77.71
8M	0.1213	2.391	0.516	2.35	0.1395	4.252	0.488	2.14
16M	0.1505	2.365	0.547	2.77	0.1502	3.992	0.479	2.60
31M	0.1543	2.665	0.608	8.75	0.1693	4.596	0.561	6.24
60M	0.1813	3.078	1.010	4.52	0.1856	5.195	1.140	7.37
120M	0.1280	2.307	0.539	4.14	0.1927	4.516	0.598	7.43

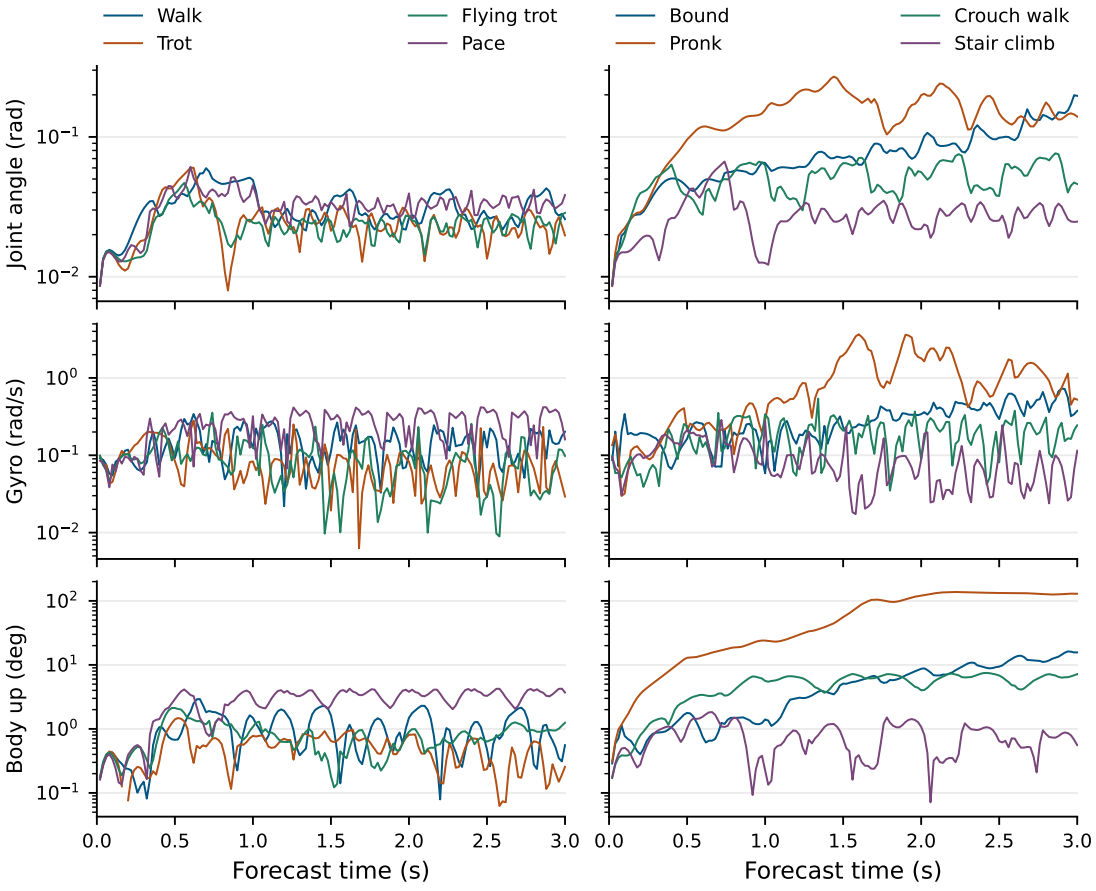


Figure 5: Simulated T2 gait replays. Errors are instantaneous across sensor channels at each 20 ms tick for the adapted 120M model against MuJoCo. Each row shares a logarithmic vertical scale across the two panels. The first two rows show channel RMSE. The last row shows the angle between predicted and simulated body-up directions.

D Sensor errors during the filmed walk

The filmed trial permits a direct comparison of the motion forecast at each tick with the onboard sensor record. Figure 6 shows the three forecasts behind Table 6. The time average of frame-wise up angles gives the table's attitude error. Joint-angle and gyro errors pool squared differences across ticks and channels before taking a square root.

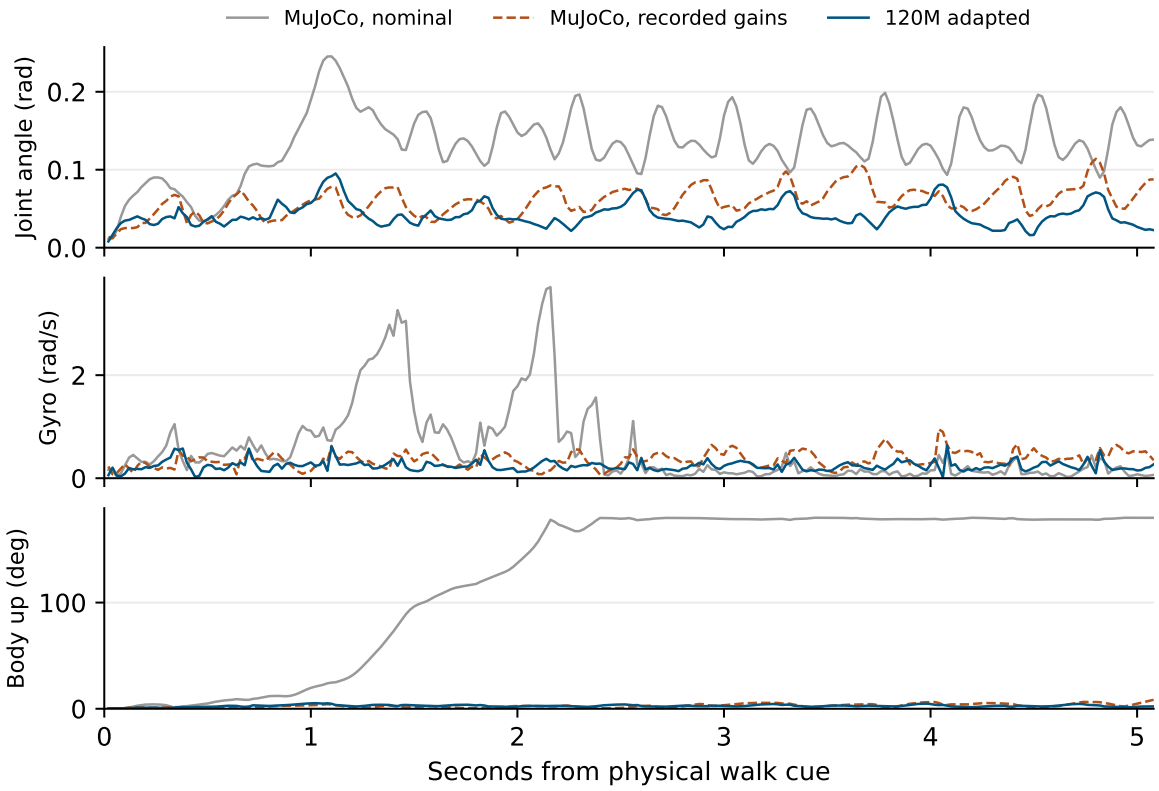


Figure 6: Real Go2, the complete 5.08 s stock-SDK walk. Each curve scores predictions against the recorded sensors under the same sequence of 254 published joint-angle targets. Switching from nominal to recorded actuator gains reduces the physics replay's attitude divergence. The travel measurements in Fig. 4 come from a separate observation of this same walk.

Numerical supplement

The source archive includes the per-shard squared errors for all 20 scaling checkpoints and the per-world errors at all 150 rollout steps for the flat-ground transfer comparison. For replay, it contains 687 model–case scores for the fixed three-second panel (11 physical cases and eight simulated cases), plus 32 model–case–horizon scores for the two named SDK trials. The filmed-run scores use the full walk. That recording and three other 0.5 m/s recordings are already members of the nine-trial SDK panel.

The self-contained file `anc/go2-results.html` indexes the cases, controllers, training records and original run diagnostics. It also gives joint-velocity, gyro and up errors for the handset and speed-selected subsets. The CSV and NumPy files in the same directory retain machine-readable scores, matched command histories and the available reference trajectories. The camera audit records calibration residuals, independent picks and endpoint sensitivity. The supplied Python scripts reproduce the figures, tables, scaling fits and paired uncertainty calculations from these numerical records.

The checkpoint family comprises 20 scan-conditioned predictors and the five projection-adapted predictors in Table 3. A forecast starts with 51 readings and the 50 joint targets between them. The first proposed target acts on the final reading. At 50 Hz, the next 150 outputs cover three seconds. The scan-conditioned family also takes one encoded terrain scan per proposed target. These aligned scans come from the recorded trajectory in the scaling study.