RoboTTT: Context Scaling for Robot Policies

Yunfan Jiang[1,2], **Yevgen Chebotar**[1], **Ruijie Zheng**[1], **Fengyuan Hu**[1], **Yunhao Ge**[1]
Jimmy Wu[1], **Tianyuan Dai**[1,3], **Scott Reed**[1], **Li Fei-Fei**[2,†], **Yuke Zhu**[1,3,†], **Linxi "Jim" Fan**[1,†]
[1]NVIDIA [2]Stanford University [3]The University of Texas at Austin [†]Equal advising
research.nvidia.com/labs/gear/robottt

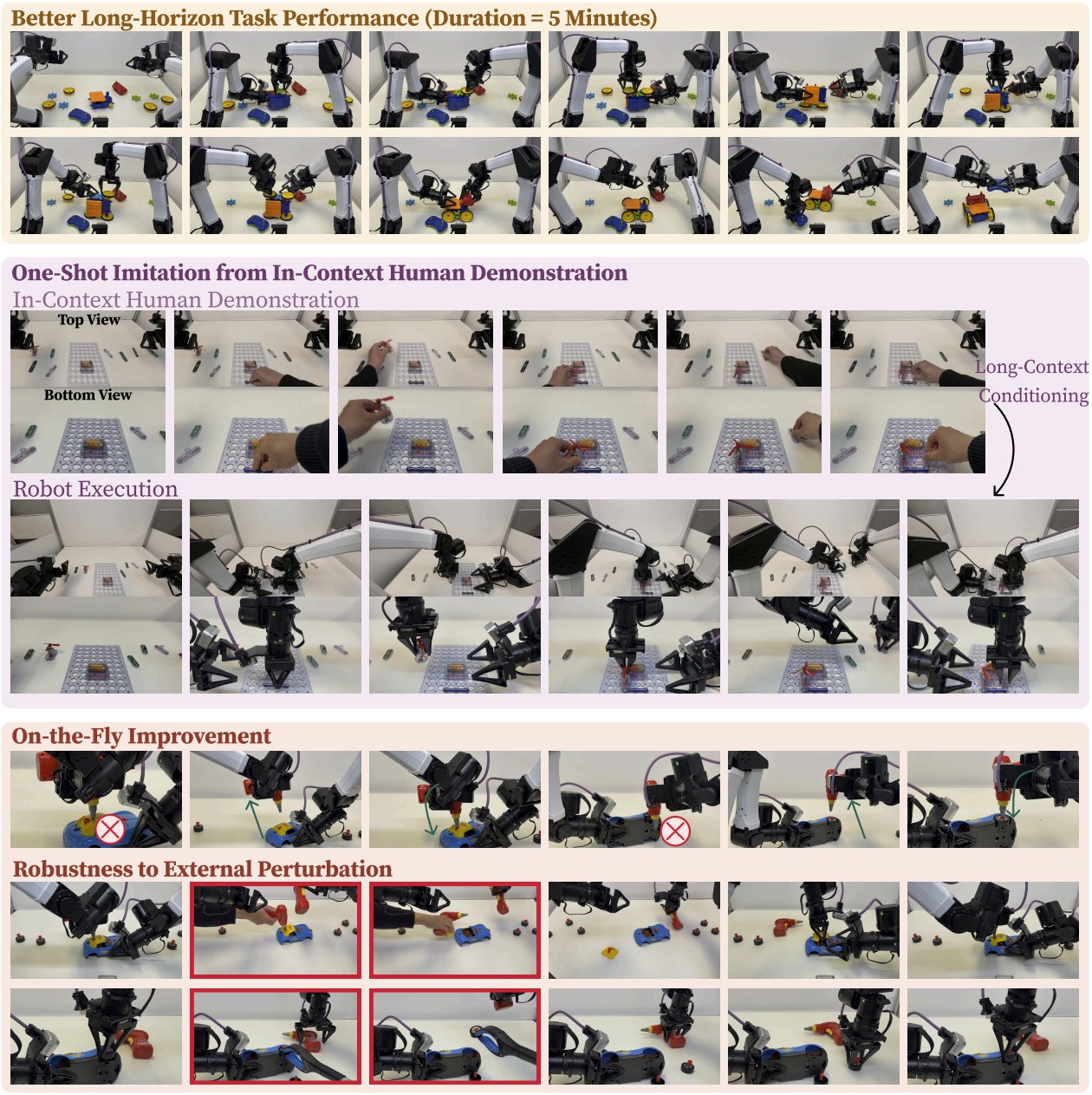

Figure 1: **RoboTTT, a long-context visuomotor policy that integrates Test-Time Training (TTT) into robot foundation models, with context scaled to 8K timesteps.** RoboTTT exhibits capabilities such as one-shot in-context imitation from human videos and on-the-fly policy improvement.

Abstract

Recent robot foundation models operate with single-step or short-history visuomotor context. We introduce *Test-Time-Training Robot Policies* (RoboTTT), a robot model and training recipe that scale visuomotor context to 8K timesteps, three orders of magnitude beyond state-of-the-art policies, without growing inference latency. At this context length, we unlock new robot capabilities: one-shot in-context imitation from human video demonstrations, on-the-fly policy improvement, robustness to perturbations, and stronger performance on multi-stage, long-horizon tasks. We also observe, for the first time, steady gains in closed-loop performance as pretraining context length scales. At its core, RoboTTT integrates Test-Time Training into robot foundation models such as Vision-Language-Action policies, yielding a sequence model whose recurrent state consists of *fast weights*, parameters updated by gradient descent during both training and inference, compressing histories into weight space and retrieving contextual information for long-context conditioning. To scale training context length, the recipe combines *sequence action forcing* with *truncated backpropagation through time*. On challenging real-robot manipulation tasks, RoboTTT improves overall performance by 87% over the single-step context baseline and fully completes a five-minute, ten-stage assembly task, which no baseline ever does. RoboTTT trained with 8K-timestep context outperforms the same model pretrained with 1K timesteps by 62%, suggesting context length as a new scaling axis for robot foundation models. Videos are available at research.nvidia.com/labs/gear/robottt**.**

Keywords: Long-Context Policies, Test-Time Training, Robot Foundation Models

1. Introduction

Most state-of-the-art robot foundation models operate with single-step or short-history visuomotor context (5; 7; 8; 29; 36; 37; 41; 47; 51; 60; 67; 75; 76). In contrast, context length has become an important scaling axis for large language models (9; 16; 27; 46; 53; 71). While longer-term reasoning in robotics can be delegated to external memory banks (50; 69), long visuomotor context remains important for capabilities such as one-shot in-context imitation from human video demonstrations (18), on-the-fly improvement from a robot's own deployment history (39), and stronger closed-loop performance on multi-stage, long-horizon tasks. This raises a natural question: *how can we build visuomotor policies that learn from and exploit arbitrarily long contexts?*

To answer this question, we introduce *Test-Time-Training Robot Policies* (*RoboTTT*, Fig. 1), a robot model and training recipe that scale visuomotor context to 8K timesteps (dubbed RoboTTT-8K), three orders of magnitude beyond state-of-the-art robot foundation models (5; 51; 75), without growing inference latency. At this context length, RoboTTT exhibits new robot capabilities. Through long-context conditioning, it performs one-shot imitation from a single in-context human video demonstration, succeeding in 6 of 10 trials while baseline methods fail entirely. It also exhibits on-the-fly policy improvement, performing 36% better than the same model not trained for this capability. Under external perturbations, it succeeds in 83% of trials versus 53% for the best short-context baseline. On multi-stage, long-horizon tasks, it improves overall task performance by 87% over the single-step context baseline and fully completes an assembly task lasting over **five minutes** and spanning **ten stages**, which no baseline ever does. Finally, we show for the first time that scaling pretraining context length yields steady gains in closed-loop performance: RoboTTT-8K achieves a 63% higher task completion score than the same model pretrained with 1K-timestep context and outperforms the best short-context baseline by 57%, suggesting context length as a new scaling axis for robot foundation models.

At its core, RoboTTT integrates Test-Time Training (TTT) (64; 78) into robot foundation models such as Vision-Language-Action (VLA) policies (51). RoboTTT is a sequence model whose recurrent state consists of *fast weights* (58): unlike *slow weights*, which are frozen at inference, fast weights are updated by gradient descent during both training and inference. This design addresses the three challenges of long-context visuomotor policies: encoding long histories with sufficient capacity, exploiting the conditioned context (12), and keeping inference cost constant in context length. First, a *fast model* (e.g., an MLP) parameterized by fast weights offers

greater capacity than the vector-valued states of recurrent neural networks. Second, training the fast model during deployment retains salient features and discards redundant ones in the dense, repetitive streams of robot observations and actions. Third, propagating fast weights over time keeps inference cost constant, whereas Transformer inference grows with history even with a KV cache. Crucially, by learning during deployment, RoboTTT enables new forms of in-context adaptation and policy improvement. For example, it conditions on a human video demonstrating a new task configuration to achieve one-shot imitation. Through *DAgger Distillation*, a training procedure that distills DAgger-style (57) failure-to-correction mappings into fast weights, it learns to improve on the fly. To scale training context length, our recipe combines *sequence action forcing* with *truncated backpropagation through time* (TBPTT), allowing context to grow without increasing GPU memory. Videos are available at `research.nvidia.com/labs/gear/robottt`.

2. Preliminaries

Test-Time Training Mechanism Test-Time Training (TTT) (64; 78) introduces *fast weights* that are updated during both training and inference to dynamically model contextual information. This contrasts with *slow weights* (i.e., model parameters), which are updated only during training and remain frozen during inference. Formally, consider a sequence of d-dimensional tokens X and its query, key, and value sequences Q, K, V, induced by projection matrices $\theta_Q, \theta_K, \theta_V$, with Q_t, K_t, V_t denoting the projections at timestep t. The fast weights W parameterize a small neural network $f_W(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$, for instance a linear layer or an MLP. At timestep t, the fast weights are updated to associate K with its corresponding value projection V through

$$W_t \leftarrow W_{t-1} - \eta \nabla_W \mathcal{L}_{\mathrm{FW}} \left(f_{W_{t-1}} \left(K_t \right), V_t \right), \tag{1}$$

where $\mathcal{L}_{\mathrm{FW}}(\hat{v}, v) = \|\hat{v} - v\|^2$ is typically a mean squared error and η denotes the (learnable) learning rate. The updated fast weights then compute the output O_t in the *apply* step via

$$O_t = f_{W_t}(Q_t). \tag{2}$$

This "update then apply" operation occurs during both training and inference. Intuitively, the update step encodes contextual information into the parameter space of the fast model f_W, and the apply step retrieves it for the downstream prediction. The projection matrices $\theta_Q, \theta_K, \theta_V$ and the fast weight initialization W_0 are learned with the outer task loss, such that the history mechanism is optimized for the task at hand. At inference, TTT thus compresses previously processed tokens into its fast weights, whereas standard full attention retains all previous keys and values in memory and attends over them at each step.

Robot Sequence Models In this work, we consider robot policies that condition on their rollout history, also known as robot sequence models (22; 32; 56; 67). Concretely, a robot trajectory $\xi = \{(o_t, q_t, A_t)\}_{t=1}^{T}$ consists of image o_t, proprioception q_t, and action chunk A_t tuples; we omit the language modality for simplicity. We learn policies $\pi(A_t | \xi_{<t}, o_t, q_t)$ that condition on the history $\xi_{<t}$ before timestep t and the current observation. For long-context policies, we aim to scale the context length $|\xi_{<t}|$.

3. Method: Test-Time-Training Robot Policies

In this section, we introduce *Test-Time-Training Robot Policies* (RoboTTT), a robot model and training recipe for learning over long-context robot trajectories. We first describe the model architecture and how we integrate TTT into modern robot foundation models, then a training recipe that combines *sequence action forcing* with *truncated backpropagation through time* (TBPTT) to scale training context length. We then describe how RoboTTT enables long-context conditioning, unlocking new forms of in-context adaptation and policy improvement: one-shot imitation from in-context human video demonstrations, and *DAgger Distillation*, a meta-learning method that teaches the policy to improve on-the-fly by distilling the DAgger-style (57) correction process, mapping suboptimal robot actions to human corrections, into its fast weights. Finally, we share implementation details.

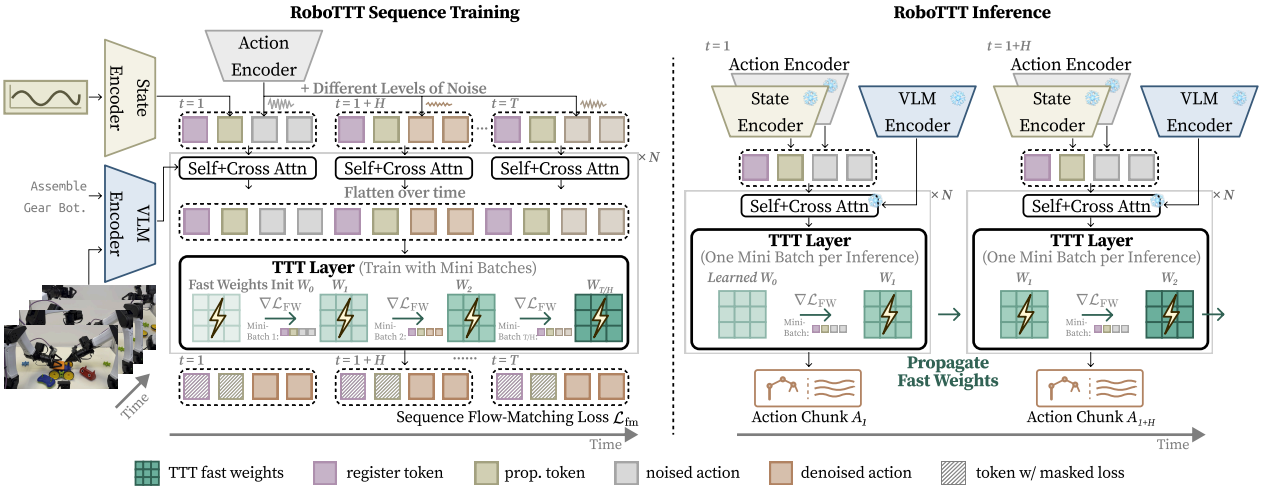

Figure 2: **RoboTTT model architecture, training, and inference.** TTT layers are added after the attention layers in the DiT action head: attention operates within each timestep, while TTT layers operate across timesteps. Training uses a sequence flow-matching loss with *sequence action forcing*, sampling the noise level independently per action chunk. Inference starts from the learned initialization W_0, updating fast weights on each observation and propagating them forward.

3.1. Model Architecture

RoboTTT integrates TTT layers into robot foundation models, such as VLA models, while preserving compatibility with the underlying architecture, and thus applies to a broad range of backbones. In this paper, we instantiate RoboTTT on top of GR00T N1.7 (51).

Test-Time Training for Robot Actions As shown in Fig. 2, RoboTTT consists of a vision-language model (VLM) backbone and a Diffusion Transformer (DiT) (52) action head with TTT layers. At timestep t, it predicts an H-step action chunk $A_t = [a_t, \ldots, a_{t+H-1}]$. We add TTT layers after the self- and cross-attention layers, so that attention processes single-step information while TTT layers process information across the time dimension. Concretely, the input to RoboTTT's DiT is a robot trajectory spanning T timesteps, $[R_1, \Phi_1, q_1, \tilde{A}_1, \ldots, R_T, \Phi_T, q_T, \tilde{A}_T]$, where Φ_t are the vision-language (VL) tokens output by the VLM, q_t is the encoded proprioception token, $\tilde{A}_t$ are the noised action tokens, and R_t are N learned register tokens (14; 30) prepended at each timestep that attend to all other tokens. Attention layers operate on the single-step tokens R_t, q_t, and $\tilde{A}_t$, and cross-attend to the VL tokens Φ_t of that timestep. The per-timestep attention outputs are then concatenated along the time dimension, $X = [R_1, q_1, \tilde{A}_1, \ldots, R_T, q_T, \tilde{A}_T]$, and passed through the TTT layers for the fast weight update (Eq. 1) and output (Eq. 2). We avoid passing the VL tokens Φ through TTT layers directly for computational efficiency, relying instead on the smaller number ($N = 16$) of register tokens R to carry VL information across time.

Gating for Preserving Pretrained Capabilities To retain the knowledge of the pretrained VLA model, RoboTTT is initialized from the base model weights and adopts a learned $\tanh$ gating mechanism (1) that keeps the contribution of TTT small at the start of training. Concretely, for each DiT layer, we learn $\alpha \in \mathbb{R}^d$, initialized to near zero (0.001), and gate the TTT contribution through

$$O = \tanh(\alpha) \odot O_{\text{TTT}} + O_{\text{attn}}, \tag{3}$$

where O_{TTT} is the TTT layer output from Eq. 2 and O_{attn} is the attention layer output. In this way, RoboTTT learns to adjust the contribution of the TTT layers without overwhelming the computation of the pretrained model.

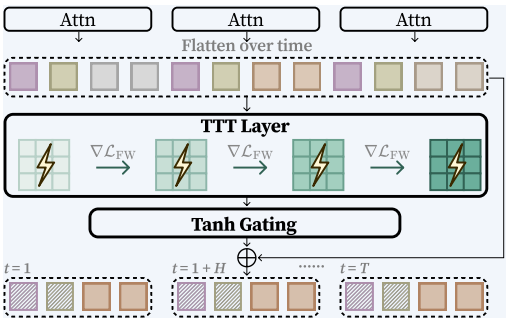

Figure 3: **Computation flow with $\tanh$ gating.** The TTT output is weighted by the learned gate $\tanh(\alpha)$ before being added to the attention output.

3.2. RoboTTT Sequence Training

We train RoboTTT on robot trajectory sequences so that fast weights are updated within each training sequence, learning both a suitable fast weight initialization and its update dynamics. The TTT projection matrices $\theta_Q, \theta_K, \theta_V$ and the fast weight initialization W_0 are learned as part of the model parameters. Specifically, given a training sequence, we run TTT over it in the inner loop with the fast weight loss $\mathcal{L}_{\text{FW}}$ (Sec. 2), compute the outer task loss at every timestep, and optimize the full model on the averaged loss. In this way, the projection matrices are learned directly from the outer task gradient, and W_0 is meta-learned through gradients of gradients (21; 65), tailoring the fast weight updates to robot trajectories.

Concretely, our dataset $\mathcal{D} = \{\xi^{(i)}\}_{i=1}^{N}$ contains N trajectories, each a sequence of language, image, proprioception, and action-chunk tuples $\xi = \{(l, o_t, q_t, A_t)\}_{t=1}^{T}$, reintroducing the language instruction l, which is shared across the trajectory. Each training sequence is a full trajectory or a contiguous sub-trajectory up to a maximum context length. Denoting the per-step flow-matching objective as ℓ_t, the sequence loss over a trajectory ξ given fast weight initialization W_0 is

$$\mathcal{L}_{\text{fm}}(\xi; W_0) = \frac{1}{T}\sum_{t=1}^{T} \ell_t\left((l, o_t, q_t, A_t); W_{t-1}\right), \tag{4}$$

where W_{t-1} is the fast weight state entering timestep t, updated to W_t inside the TTT layers (Eq. 1). Each optimization step then takes a gradient step w.r.t. $\mathcal{L}_{\text{fm}}$, updating both regular model weights and the fast weight initialization.

Sequence Action Forcing RoboTTT is trained with a flow-matching objective for actions: the DiT action head learns to denoise $\tilde{A}_t = A_t^\tau = \tau A_t + (1-\tau)\epsilon$, where $\tau \in [0, 1]$ is the flow-matching timestep and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the sampled noise. In sequence training, we find it necessary to sample the noise level *independently* for each action chunk in the sequence, a technique we call *sequence action forcing*. Without it, training is unstable, possibly because sharing one noise level across the sequence (full-sequence diffusion) makes entire sequences uniformly easy (low noise) or uniformly hard (high noise) to learn, echoing the findings of Chen et al. (10).

To summarize, denoting the DiT action head as v_θ and the timestep-t tuple as $\xi_t = (l, o_t, q_t, A_t)$, we train RoboTTT with sequence action forcing by minimizing

$$\mathcal{L}_{\text{fm}}(\xi; W_0) = \frac{1}{T}\sum_{t=1}^{T} \ell_t(\xi_t; W_{t-1}) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\tau_t, \epsilon}\left[\left\| v_\theta\left(\Phi_t, A_t^{\tau_t}, q_t; W_{t-1}\right) - (A_t - \epsilon)\right\|^2\right], \tag{5}$$

where each τ_t is sampled independently as $\tau_t = s(1-u)$, $u \sim \text{Beta}(1.5, 1)$, $s = 0.999$.

Truncated Backpropagation Through Time Training on long sequences with full backpropagation through time (BPTT) stores activations for every timestep, so GPU memory grows with sequence length. We instead adopt truncated backpropagation through time (TBPTT): the input sequence is divided into segments, and gradients flow only within each segment (Fig. 4). Crucially, the fast weights are carried over across segment boundaries, so TTT continues over the entire sequence, while their gradients are detached at the boundaries. GPU memory is thus determined by the segment length rather than the total sequence length, allowing arbitrarily long training contexts under a fixed memory budget. Note that the fast weight initialization W_0 still receives gradients through the first segment, whose updates originate directly from W_0.

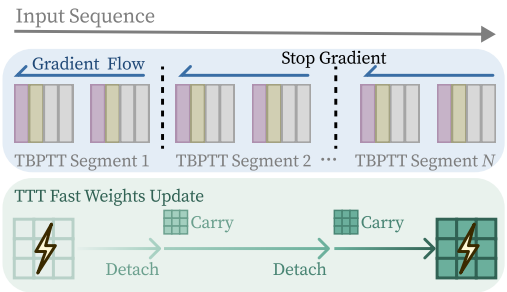

Figure 4: **TBPTT.** Gradients are truncated at segment boundaries; fast weights carry over, so TTT continues over the entire sequence.

Inference As illustrated in Fig. 2, RoboTTT starts each rollout from the learned initialization W_0, updates the fast weights on the current observation, and propagates them to the next timestep. At each timestep, action chunks are generated with k-step denoising.

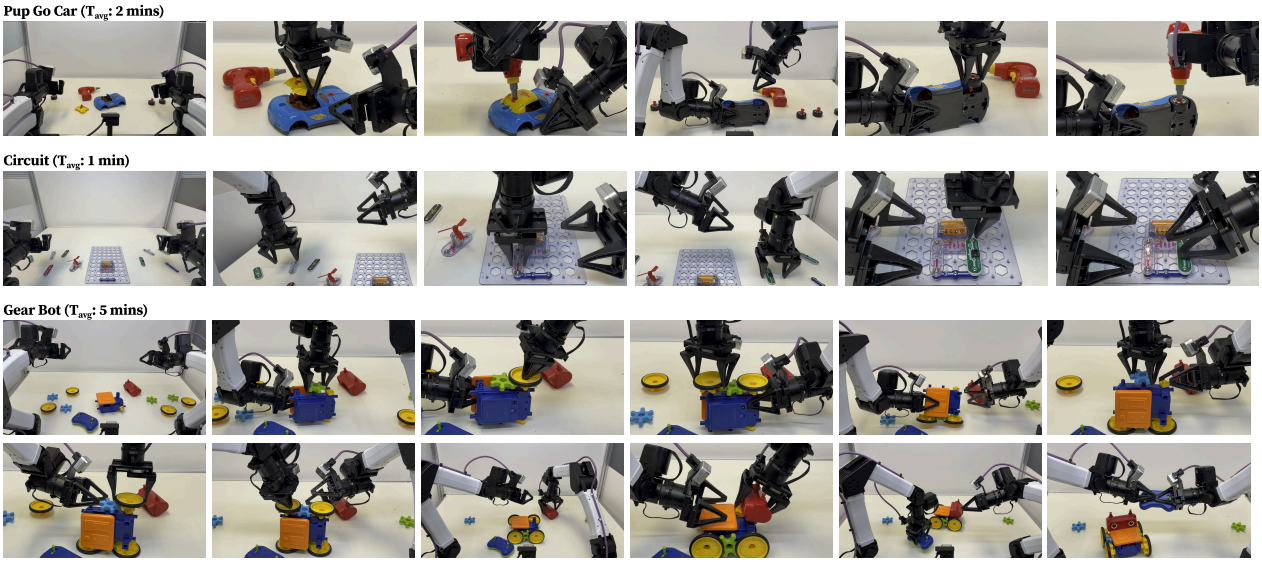

Figure 5: **Evaluation tasks.** Three long-horizon assembly tasks on a YAM bimanual setup; each row shows one rollout. **Top, Pup Go Car:** toy vehicle assembly, 2-minute average episodes. **Middle, Circuit:** target configuration specified by a language prompt or a one-shot human video demonstration; 1-minute episodes. **Bottom, Gear Bot:** ten-stage assembly spanning five minutes, our longest-horizon task.

3.3. Effective Learning from Context

RoboTTT decouples fast weight updates from slow weight updates: by masking the flow-matching loss on selected timesteps, those timesteps serve as pure context, updating the fast weights without providing an imitation target. This flexibility lets RoboTTT learn from heterogeneous contexts, such as human video demonstrations or the robot's own suboptimal rollouts.

Imitation from In-Context Video Demonstrations We pair human video demonstration sequences ξ^{video} with robot trajectories ξ^{robot} of the same task: the video sequence updates the fast weights only (its flow-matching loss is masked), while the action loss is computed on the robot trajectory conditioned on the updated fast weights. Trained on such pairs, the model learns to extract task information from the in-context video; at test time, conditioning on a single human video of an unseen task configuration yields one-shot imitation.

DAgger Distillation Consider a robot rollout with human corrections collected as in DAgger (57): whenever the robot makes a mistake, a human operator intervenes with corrective actions, yielding a trajectory $\xi^{\text{DAgger}} = \{(l, o_t, q_t, A_t)\}_{t=1}^{T}$ in which each executed action chunk A_t is either a robot action A_t^{R} or a human correction A_t^{H}. Such interleaved rollouts trace a natural pattern of online policy improvement. Standard DAgger fine-tunes on the human corrections and discards the suboptimal robot actions; yet it is precisely these actions that reveal what failure each correction responds to. RoboTTT instead uses both, in asymmetric roles: during sequence training, the fast weights are updated on the full interaction history, not only the executed corrections but also the suboptimal robot actions themselves, while the flow-matching loss is masked to the human corrections only.

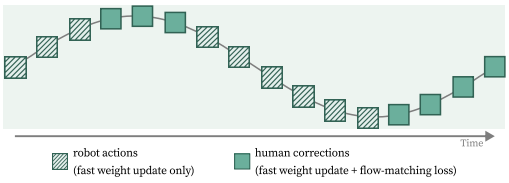

Figure 6: **DAgger Distillation.** During sequence training, all executed actions update the fast weights, but the flow-matching loss is computed only on human corrections.

We call this procedure *DAgger Distillation* (Fig. 6). This asymmetry, failures as context and corrections as targets, is precisely how the human's failure-to-correction mapping is distilled into the fast weights: the model learns to produce corrections in response to failures, rather than to imitate corrections in isolation, making fuller use of the same collected data. We view it as an instantiation of Algorithm Distillation (39) in robotics:

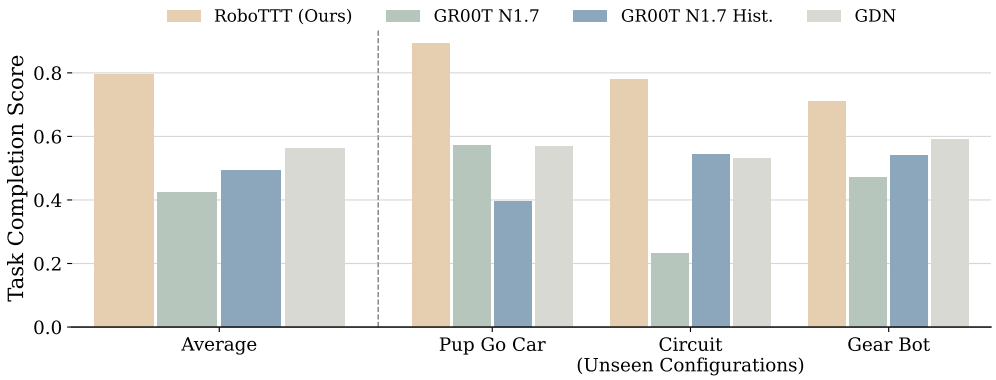

Figure 7: **Main evaluation: task completion scores on three assembly tasks.** Scores are rubric-based, reported in percent; higher is better.

the improvement process induced by human interventions is distilled into the policy's fast-weight adaptation. At test time, the model performs such corrections online, without human intervention; its own corrections then enter the history and are absorbed into the fast weights, exactly as the human corrections were during training. As we show in Sec. 4, DAgger Distillation yields stronger failure recovery and higher task performance than standard DAgger training.

3.4. Implementation Details

We instantiate RoboTTT on pretrained GR00T N1.7 (51), adding a TTT layer to each of its 16 DiT layers; each fast model is a two-layer MLP. We pretrain on a mixture of tabletop bimanual robot data and egocentric human data (81), gradually increasing the pretraining context length up to the target (e.g., 8K timesteps for RoboTTT-8K), for 30K steps on 16 NVIDIA GB200 GPUs. We then post-train on each downstream task at 1K context length for 20K steps. Further details are in Appendix A.

4. Experiments

Experiment Settings We evaluate on three long-horizon, multi-stage assembly tasks requiring bimanual manipulation and dexterity (Fig. 5): Pup Go Car, Circuit, and Gear Bot. All experiments use a YAM bimanual setup with four RGB cameras: top, bottom, left wrist, and right wrist. For each task, we collect 8, 6, and 5 hours of real-robot data, with average episode lengths of 2 minutes, 1 minute, and 5 minutes, respectively. The Circuit task has 80 assembly configurations, varying components, assembly order, and number of pieces; the target configuration is specified through the language prompt (or a one-shot human video demonstration). We train on 20 configurations and test on the remaining 60.

We compare RoboTTT against three baselines: **GR00T N1.7** (51) with single-step context; **GR00T N1.7 Hist.**, GR00T N1.7 with one history frame; and **GDN**, in which the TTT layers of RoboTTT are replaced with Gated DeltaNet layers (74), a linear-complexity recurrent memory that updates its state without test-time gradient descent. All methods are post-trained on the same task data: sequence models use a 1K-timestep context, and non-sequence models are trained to a matched compute budget. Each policy is evaluated for 20 trials (10 for Gear Bot due to its substantially longer horizon) across varied configurations. We report the number of fully successful trials and a rubric-based task completion score normalized to $[0, 1]$ (reported in percent). Hyperparameters and score rubrics are in Appendix B.

RoboTTT consistently outperforms baselines on dexterous, long-horizon tasks. As shown in Fig. 7 and Table 1, RoboTTT achieves an average task completion score of 79%, 87% higher than the single-step context baseline GR00T N1.7 (42%) and 41% higher than the best baseline GDN (56%), with the most fully successful trials on every task. Notably, on Gear Bot, which requires five minutes on average for full completion, RoboTTT

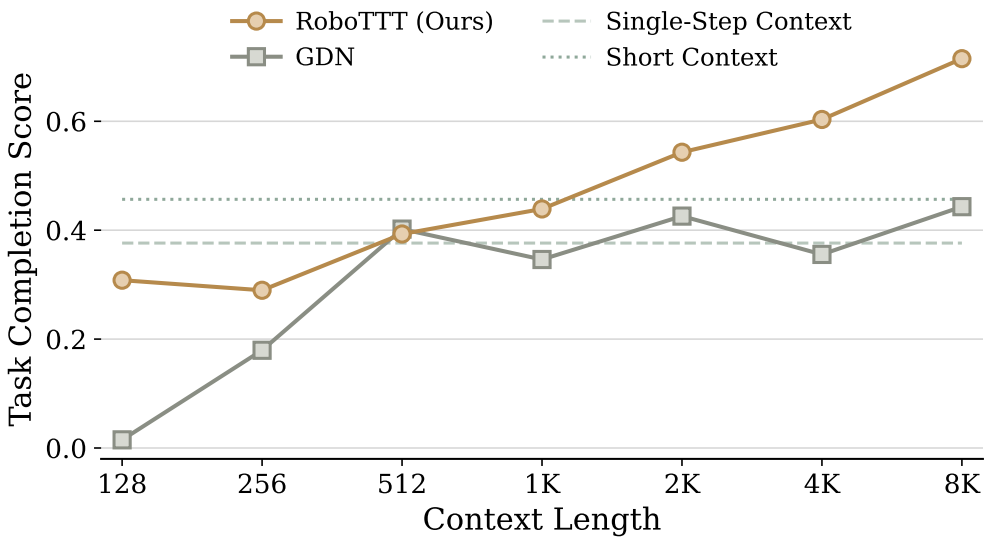

Figure 8: **Closed-loop performance scales with pretraining context length.** Average task completion score across the three tasks as pretraining context length grows from 128 to 8K timesteps. RoboTTT improves steadily, surpassing the best short-context baseline from 1K onward with no sign of saturation; GDN does not benefit from longer context. *Single-Step Context* and *Short Context* denote GR00T N1.7 and GR00T N1.7 Hist.. All evaluations in this figure predate the DAgger training used for Pup Go Car in the main results.

achieves full successes (2 of 10) while no baseline ever does. Qualitatively, RoboTTT excels in three aspects. First, it tracks task progress: in multi-stage assembly, visually similar stages cause state aliasing, leading baselines to perform wrong motions or skip stages, whereas the fast weights, updated on the fly, retain salient features of the history and disambiguate the current stage. Second, it exhibits *strategic recovery*: during roof drilling on Pup Go Car, if the drill misses the roof screw, RoboTTT raises the arm, re-aligns, and re-attempts, while baselines proceed to the next stage as if the previous one had succeeded. Third, it is more precise in fine-grained stages such as inserting and snapping circuit components; we attribute this to long context mitigating partial observability, as past observations of the object of interest inform actions when it is currently occluded. The relevant observation window is difficult to specify a priori; RoboTTT instead learns what to retain, suggesting that with a sufficiently expressive sequence model, the use of long context can be learned rather than hand-designed.

Can history alone match RoboTTT? Naively concatenating past observations does not reliably help: on Pup Go Car, GR00T N1.7 Hist. scores 39.5% versus 57% for its no-history counterpart GR00T N1.7, as appended histories can introduce spurious correlations and leave the robot temporally out of distribution at inference. GDN, which compresses history into a recurrent state, improves over GR00T N1.7 on Circuit and Gear Bot but not Pup Go Car. Notably, both GDN and RoboTTT maintain fixed-size states; the difference is the update rule. We hypothesize that the gated delta rule, a linear associative update without test-time gradient descent, struggles to extract structure from dense, repetitive robot streams across thousands of timesteps, whereas RoboTTT's nonlinear fast model, updated by gradient descent at test time, is the more expressive compressor.

Method	Pup Go Car	Circuit	Gear Bot
RoboTTT	**9 / 20**	**13 / 20**	**2 / 10**
GR00T N1.7	3 / 20	3 / 20	0 / 10
GR00T N1.7 Hist.	0 / 20	8 / 20	0 / 10
GDN	3 / 20	8 / 20	0 / 10

Table 1: **Main evaluation: fully successful trials on three assembly tasks**, out of 20 per task (10 for Gear Bot). RoboTTT is the only method with full successes on Gear Bot.

Scaling pretraining context length yields steady gains in closed-loop performance. We pretrain both RoboTTT and GDN at context lengths from 128 timesteps (a few seconds) to 8K (over four minutes), then post-train and evaluate closed-loop performance on all three tasks (Fig. 8). As references, *Single-Step Context* denotes GR00T N1.7, conditioned on the current observation only, and *Short Context* denotes GR00T N1.7 Hist., with one additional history frame. RoboTTT exhibits a clear scaling trend: closed-loop performance increases

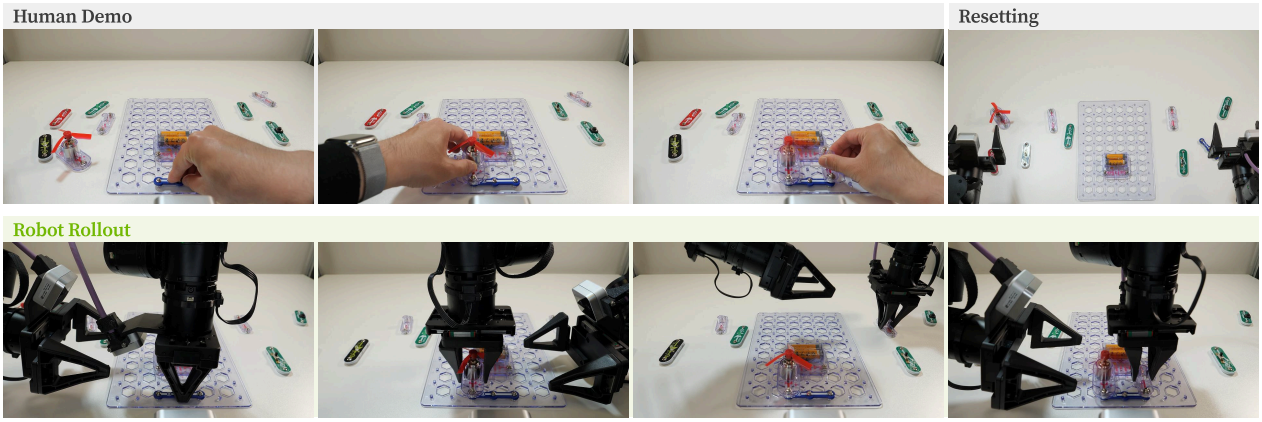

Figure 9: **One-shot imitation from an in-context human video.** Continuous footage: a human demonstrates an unseen configuration (frames 1–3), the scene is reset (frame 4), and RoboTTT reproduces the assembly (row 2). The prompt is identical across configurations, so the target is identifiable only from the video.

steadily with pretraining context length, reaching 71.5% at 8K, 63% higher than the same model pretrained at 1K (43.9%) and 57% higher than the best short-context baseline GR00T N1.7 Hist. (45.6%), with no sign of saturation. GDN shows no such trend. We attribute the difference to the two update rules: RoboTTT's fast weights are updated by gradient descent, and the outer loss meta-learns their initialization W_0 and update dynamics, which longer training sequences shape over more update steps; GDN's linear associative state admits no such meta-learning. Below 1K, RoboTTT remains competitive (still outperforming GDN at matched lengths) but falls short of its longer-context variants. We attribute this to the rollout horizon exceeding the training context: 1K timesteps is about half a minute, shorter than our shortest task episode, so at inference the fast weights are updated far beyond the window seen in training, and positional embeddings extend to unseen positions.

Long-context conditioning unlocks one-shot imitation and perturbation robustness. We first evaluate one-shot imitation. For the Circuit task, we collect additional human demonstration videos in which the robot stays idle and a human assembles the circuit by hand. For each assembly configuration in the training set, we collect 5–20 human videos with varied initial layouts. During training, we sample a human video and a robot trajectory of the same configuration and concatenate them into a single training sequence, masking the flow-matching loss on the video portion (Sec. 3.3). We use the same task prompt, "assemble circuit," for all configurations, so the target configuration is identifiable only from the in-context video. Since GR00T N1.7 cannot condition on context and human videos far exceed GR00T N1.7 Hist.'s history window, we compare against GDN.

As shown in Table 2, RoboTTT follows the in-context demonstration (Fig. 9) and achieves six successful assemblies out of ten trials, while GDN fails entirely, picking up wrong components or assembling in the wrong order. This suggests that although recurrent-memory policies can encode contextual information, they struggle to use it; RoboTTT instead queries a fast model updated by gradient descent on the history, retrieving the demonstrated configuration when needed.

	Task Completion Score	Number of Successful Rollouts
RoboTTT	**65%**	**6 / 10**
GDN	33%	0 / 10

Table 2: **One-shot imitation on Circuit.** Task completion score and fully successful trials, conditioning on one in-context human video of an unseen configuration.

We next evaluate perturbation robustness: beyond conditioning across episodes, RoboTTT also conditions *within* an episode. On Pup Go Car, a human removes the yellow roof after the robot installs it, or removes a tire after insertion; a policy that conditions on its own rollout should return to the pre-perturbation stage and reinstall the part. We collect 30 minutes of perturbation data and co-train it with the task data. As shown in Table 3, all methods exhibit some robustness, likely from this co-training, but the long-context methods react suc-

Method	Roof Perturbation	Tire Perturbation
RoboTTT	**15 / 20**	**18 / 20**
GR00T N1.7	10 / 20	11 / 20
GR00T N1.7 Hist.	3 / 20	5 / 20
GDN	13 / 20	**18 / 20**

Table 3: **Robustness to external perturbations.** Each entry reports successful recoveries, i.e., where the policy reassembled the removed part, out of 20 per condition.

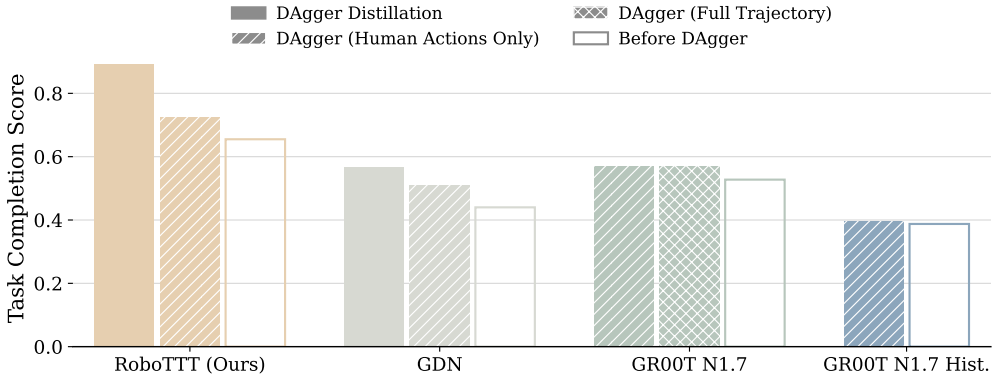

Figure 10: **DAgger Distillation results on Pup Go Car.** Task completion after fine-tuning on a pool of 100 DAgger trajectories (50 collected with RoboTTT, 50 with GR00T N1.7). *DAgger Distillation* applies to the sequence models RoboTTT and GDN.

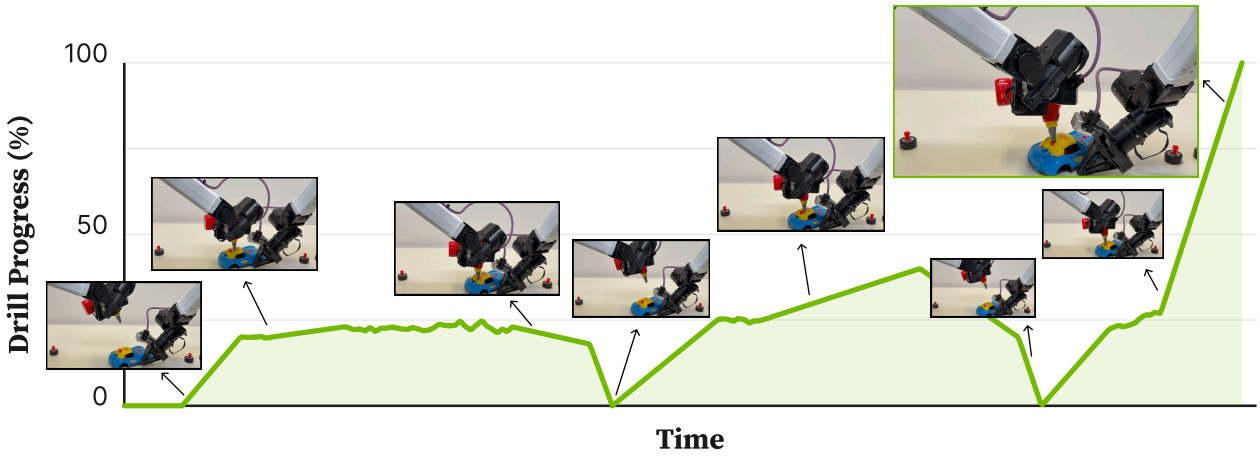

Figure 11: **On-the-fly recovery learned through DAgger Distillation.** While tightening the roof screw on Pup Go Car, RoboTTT misses the screw (frames 1–3), raises the arm and re-attempts, getting closer but missing again (frames 4–5), then re-adjusts once more and succeeds (frames 6–8).

cessfully more often: RoboTTT recovers in 15/20 roof-perturbation trials versus 13/20 for GDN and at most 10/20 for the short-context baselines, and both RoboTTT and GDN recover in 18/20 tire trials. This supports the view that visuomotor context improves within-episode conditioning; example recoveries are shown in Fig. 1.

RoboTTT learns to recover on the fly and improves upon standard DAgger. On Pup Go Car, we study *DAgger Distillation* against alternative uses of DAgger data (Fig. 10). We collect 50 DAgger trajectories each with RoboTTT and GR00T N1.7 as the base policy, and use the pooled 100 trajectories to train all methods. Standard DAgger, fine-tuned on human corrections alone, improves the base policies by 9% on average across the four methods, and by 13% on the two sequence models. DAgger Distillation, applicable to the sequence models RoboTTT and GDN, takes the full trajectory, including the suboptimal robot actions, as context while computing the imitation loss only on the human corrections (Sec. 3.3). From the same DAgger data, it yields a 33% average improvement: 36% for RoboTTT and 29% for GDN. Notably, the suboptimal robot actions carry no value as imitation targets: fine-tuning GR00T N1.7 on the full trajectories, robot actions included, performs identically to corrections alone (57% for both). Their value is as context: qualitatively, most of DAgger Distillation's gain comes from recovering after wrong actions (Fig. 11), indicating that the failure-to-correction mappings distilled into the fast weights manifest as on-the-fly improvement during rollout. That GDN also improves substantially suggests DAgger Distillation applies generally to sequence-model policies, though RoboTTT benefits the most.

Design choices that matter for RoboTTT's performance. We ablate key design choices. First, we compare against two variants: RoboTTT without sequence action forcing (*No Seq. Action Forcing*) and RoboTTT with the fast model replaced by a linear layer (*TTT Linear*). Second, we trace the development of RoboTTT as a roadmap: TTT layers processing only state tokens (*State Tokens*), then additionally passing action tokens through the TTT layers (+ *Action Tokens*), and finally inserting learned register tokens (+ *Register Tokens*), which yields the full RoboTTT. Since the register tokens add capacity independent of TTT, we also augment GR00T N1.7 with the same number of register tokens, so the comparison isolates the TTT mechanism from the extra tokens.

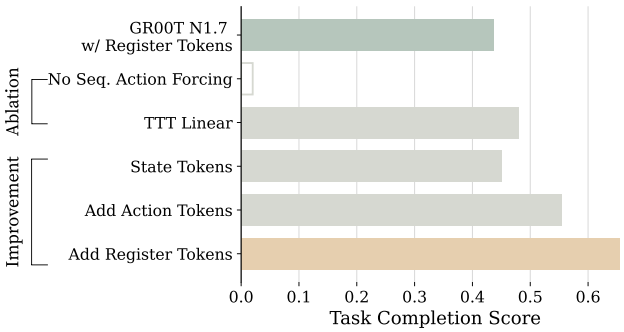

Figure 12: **Ablation results on the Pup Go Car task.** We ablate sequence action forcing and the fast-model architecture (MLP vs. linear), and trace each component added during development (state tokens, action tokens, register tokens). GR00T N1.7 with register tokens is included for a matched-token comparison.

As shown in Fig. 12, removing sequence action forcing during training significantly hurts closed-loop performance: the resulting inaccurate motions leave the robot unable to make meaningful progress. This underscores the importance of applying different noise levels across noised action chunks during sequence training. The TTT-linear variant outperforms the GR00T N1.7 baseline but remains suboptimal, 27% worse than the MLP fast model, suggesting that expressive, nonlinear fast models matter most, echoing findings in vision and language modeling tasks (78).

Starting from a TTT MLP processing only state tokens, we improve performance by gradually adding tokens. Adding action tokens yields a 23% relative improvement: aware of its past actions, the model better captures environment dynamics. Adding register tokens yields a further 18% relative improvement. In contrast, register tokens do not help GR00T N1.7. This suggests that learned register tokens are beneficial only when paired with TTT's temporal modeling, helping the model encode contextual information.

5. Related Work

Long-Context Policies Most state-of-the-art robot foundation models operate with single-step or short-history context. Most Vision-Language-Action models (VLAs) and some World Action Models (WAMs) take only the current observation (5; 29; 36; 37; 51; 60; 76) or a few consecutive observations (typically 2 to 8) (7; 8; 41; 47; 67; 75). Some work extends the observation window by, e.g., visually supplying movement trajectories (80), compressing vision-language tokens (31), predicting past actions (68), or caching and gating history tokens (23). These models nonetheless remain limited to a fixed context size. For longer horizons, history can be delegated to higher-level semantics such as keyframes (50; 61) and language (44; 69), but long visuomotor context remains important for capabilities such as in-context learning (18; 22), on-the-fly policy improvement (39), and adaptation (38; 55; 66).

Another line of work processes the entire rollout history autoregressively (6; 22; 32; 40; 56). While these models capture long-context dependencies well, they are computationally prohibitive for real-robot deployment over long horizons, since decoding latency with a KV cache grows linearly with context length. Recurrent neural network (RNN) policies (49) offer an alternative with constant inference complexity, but traditional architectures such as LSTMs (26) scale worse than their full-attention counterparts (34). RoboTTT can also be viewed as an RNN policy whose recurrent states are fast weights, updated by gradient descent on the Test-Time Training (TTT) loss. Crucially, this recurrent structure lets us scale context length, and we show that RoboTTT consistently outperforms the best single-step and short-context baselines once context length is sufficiently scaled.

Building long-context policies requires addressing the spurious correlations introduced by history (15; 70), where policies overfit to past actions implicitly encoded in past observations. Prior work mitigates this

by summarizing context histories (50; 61), introducing auxiliary objectives (68), or selectively bypassing context (23). RoboTTT instead addresses it through the learned TTT fast weights, which dynamically encode relevant information into the parameter space while erasing redundant features.

Test-Time Training Test-Time Training (TTT) (63) is a paradigm in which neural networks rapidly update a small subset of their parameters, known as *fast weights*, using a self-supervised objective during both training and inference, enabling continuous storage and retrieval of contextual information. Recent work has developed improved test-time optimization and online learning objectives (3; 4; 35; 78), tighter architectural integration with language and vision models (20; 24; 65; 79), and more efficient training strategies (42; 43). First demonstrated for language modeling (77), TTT has since shown promising results in video generation (13), computer vision (24), and 3D reconstruction (11). These modalities are inherently sequential, and TTT's success there suggests strong potential for robotics, where agents likewise interact with their environments in a continuous, streaming manner.

While several recent works (2; 45; 82) use the term "test-time training" for robotics, they do not employ fast weights, instead collecting extra data on test tasks to fine-tune the entire model. The closest setting to ours is Ziakas and Russo (83), which equips Vision-Language Models (VLMs) with fast weights to adapt value functions for robotic tasks. RoboTTT instead builds robot visuomotor policies on top of TTT layers. By scaling the training context length, we show that RoboTTT exhibits new capabilities such as one-shot imitation from in-context human demonstration videos and on-the-fly policy improvement, while achieving stronger closed-loop performance on long-horizon tasks.

Robot Foundation Models Recent years have seen rapid progress in robot foundation models that build on large-scale vision-language pretraining and adapt it to robotic control (5; 7; 8; 19; 29; 32; 36; 51; 67; 72; 80). A common paradigm initializes from pretrained VLMs and adds an action generation module mapping multimodal representations to robot actions. Existing approaches differ primarily in how actions are represented and predicted. One line formulates control as autoregressive sequence modeling, discretizing actions into tokens generated by next-token prediction (36; 54; 72). Another preserves continuous action spaces by coupling pretrained VLMs with diffusion or flow-matching policy heads (5; 17; 51; 67), modeling multimodal action distributions more expressively. While RoboTTT can in principle be a plug-and-play module for any robot foundation model architecture, here we instantiate it on the flow-matching GR00T-N1.7 (51) policy, our default backbone throughout. Notably, although GR00T-N1.7 was trained with only single-step or short context, RoboTTT scales its context to 8K timesteps (about five minutes at 30 Hz control), and we find that capabilities such as long-context conditioning emerge only once context length is sufficiently scaled.

6. Limitations and Conclusion

This work has a few limitations. First, scaling training context length increases training cost; future work could adopt more recent TTT training techniques such as TNT (42). Second, while we develop a principled way of integrating TTT into robot foundation models, future work might explore robotics-oriented objectives for the TTT layers (as explored for vision (24)). Finally, although RoboTTT improves task performance substantially, it does not handle every failure mode encountered in deployment; combining it with reinforcement learning to optimize task success directly is a natural next step.

We present RoboTTT, a robot model and training recipe that scale the visuomotor context of robot policies to 8K timesteps. At this context length, RoboTTT unlocks new robot capabilities: one-shot imitation from in-context human video demonstrations, on-the-fly policy improvement, robustness to external perturbations, and stronger closed-loop performance on multi-stage, long-horizon tasks. At its core, RoboTTT integrates Test-Time Training into robot foundation models as the sequence modeling mechanism along the time dimension, and its training recipe, combining sequence action forcing with truncated backpropagation through time, makes training on long sequences tractable. We observe for the first time that scaling pretraining context length yields steady gains in closed-loop performance, suggesting context length as a new scaling axis for robot foundation models.

Acknowledgments

We thank Frederik Ebert, Letian Fu, Abhishek Gupta, Joel Jang, Hanjung Kim, Dantong Niu, Rutav Shah, You Liang Tan, Josiah Wong, Haoyu Xiong, Ruohan Zhang, Tianyuan Zhang, and Chuning Zhu for constructive discussions. We thank Zhe Zhang and Connor Pedersen for compute cluster support. We thank Amy Nguyen, Matin Furutan, Matin Nikoui, Mona Abbas, Lion Park, Ramanpreet Singh, Alaa Eltayeb, Dona Alhabibi, Marcelo Kulik, Nachiket Timmanagoudar, and Josiah Minor for real-robot operation, data collection, and evaluation. We thank Tri Cao and Yuqi Xie for help with the release. Last but not least, we thank the NVIDIA GEAR Team and the Stanford Vision and Learning Lab for their continuous support.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv: 2204.14198*, 2022. 4

[2] Zechen Bai, Chen Gao, and Mike Zheng Shou. Evolve-vla: Test-time training from environment feedback for vision-language-action models. *arXiv preprint arXiv: 2512.14666*, 2025. 12

[3] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv: 2501.00663*, 2024. 12

[4] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv: 2504.13173*, 2025. 12

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π0: A vision-language-action flow model for general robot control, 2024. *URL https://arxiv. org/abs/2410.24164*, 2024. 2, 11, 12

[6] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauzá Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, A. Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, M. F. Martins, Rugile Pevceviciute, M. Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Zolna, Scott E. Reed, Sergio Gómez Colmenarejo, Jon Scholz, A. Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg O. Sushkov, Thomas Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, M. Riedmiller, Jost Tobias Springenberg, R. Hadsell, F. Nori, and N. Heess. Robocat: A self-improving generalist agent for robotic manipulation. *Trans. Mach. Learn. Res.*, 2024. 11

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, K. Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, A. Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, S. Levine, Yao Lu, U. Malla, D. Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, M. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, S. Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Q. Vuong, F. Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *Robotics: Science and Systems*, 2022. doi: 10.48550/arXiv.2212.06817. URL https://arxiv.org/abs/2212.06817v2. 2, 11, 12

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, K. Choromanski, Tianli Ding, Danny Driess, Kumar Avinava Dubey, Chelsea Finn, Peter R. Florence, Chuyuan Fu, Montse Gonzalez Arenas, K. Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, A. Irpan,

Nikhil J. Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, S. Levine, H. Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, M. Ryoo, Grecia Salazar, Pannag R. Sanketi, P. Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Q. Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Ted Xiao, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *Conference on Robot Learning*, 2023. doi: 10.48550/arXiv.2307.15818. URL https://arxiv.org/abs/2307.15818v1. 2, 11, 12

[9] 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. *arXiv preprint arXiv: 2005.14165*, 2020. 2

[10] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *arXiv preprint arXiv: 2407.01392*, 2024. 5

[11] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training, 2026. URL https://arxiv.org/abs/2509.26645. 12

[12] Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. Robomme: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026. 2

[13] Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17702–17711, June 2025. 12

[14] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1. 4

[15] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/947018640bf36a2bb609d3557a285329-Paper.pdf. 11

[16] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending LLM context window beyond 2 million tokens. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 11091–11104. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/ding24i.html. 2

[17] Danny Driess, Jost Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026. 12

[18] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017. 2, 11

[19] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact2: Action reasoning models for real-world deployment, 2026. URL https://arxiv.org/abs/2605.02881. 12

[20] Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dTWfCLSoyl. 12

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv: 1703.03400*, 2017. 5

[22] Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024. 3, 11

[23] Yihuai Gao, Jinyun Liu, Shuang Li, and Shuran Song. Gated memory policy. *arXiv preprint arXiv: 2604.18933*, 2026. 11, 12

[24] Dongchen Han, Yining Li, Tianyu Li, Zixuan Cao, Ziming Wang, Jun Song, Yu Cheng, Bo Zheng, and Gao Huang. Vit^3: Unlocking test-time training in vision. *arXiv preprint arXiv: 2512.01643*, 2025. 12

[25] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv: 1606.08415*, 2016. 20

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735. 11

[27] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv: 2404.06654*, 2024. 2

[28] 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. *arXiv preprint arXiv: 2404.06395*, 2024. 20

[29] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054. 2, 11, 12

[30] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv: 2103.03206*, 2021. 4

[31] Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv: 2510.04246*, 2025. 11

[32] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv: 2210.03094*, 2022. 3, 11, 12

[33] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. 20

[34] 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 preprint arXiv: 2001.08361*, 2020. 11

[35] Mahdi Karami, Razvan Pascanu, and Vahab Mirrokni. Lattice: Learning to efficiently compress the memory. *arXiv preprint arXiv: 2504.05646*, 2025. 12

[36] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ZMnD6QZAE6. 2, 11, 12

[37] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv: 2601.16163*, 2026. 2, 11

[38] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021. 11

[39] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Stenberg Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=hy0a5MMPUv. 2, 6, 11

[40] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv: 2601.21998*, 2026. 11

[41] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv: 2503.00200*, 2025. 2, 11

[42] Zeman Li, Ali Behrouz, Yuan Deng, Peilin Zhong, Praneeth Kacham, Mahdi Karami, Meisam Razaviyayn, and Vahab Mirrokni. Tnt: Improving chunkwise training for test-time memorization. *arXiv preprint arXiv:2511.07343*, 2025. 12

[43] Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=E34AlVLN0v. 12

[44] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025. 11

[45] Changyu Liu, Yiyang Liu, Taowen Wang, Qiao Zhuang, James Chenhao Liang, Wenhao Yang, Renjing Xu, Qifan Wang, Dongfang Liu, and Cheng Han. On-the-fly vla adaptation via test-time reinforcement learning. *arXiv preprint arXiv:2601.06748*, 2026. 12

[46] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024. 2

[47] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=yAzN4tz7oI. 2, 11

[48] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2017. 20

[49] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, S. Savarese, Yuke Zhu, and Roberto Mart'in-Mart'in. What matters in learning from offline human demonstrations for robot manipulation. *Conference on Robot Learning*, 2021. 11

[50] Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. Bpp: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv: 2602.15010*, 2026. 2, 11, 12

[51] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv: 2503.14734*, 2025. 2, 4, 7, 11, 12, 20, 22

[52] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv: 2212.09748*, 2022. 4, 20

[53] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u. 2

[54] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747. 12

[55] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023. 11

[56] Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=1ikK0kHjvj. 3, 11

[57] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html. 3, 6

[58] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021. 2

[59] Min Shi, Fuxiao Liu, Shihao Wang, Shijia Liao, Subhashree Radhakrishnan, De-An Huang, Hongxu Yin, Karan Sapra, Yaser Yacoob, Humphrey Shi, Bryan Catanzaro, Andrew Tao, Jan Kautz, Zhiding Yu, and Guilin Liu. Eagle: Exploring the design space for multimodal llms with mixture of encoders. In *ICLR*, 2025. 20

[60] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv: 2506.01844*, 2025. 2, 11

[61] Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv: 2510.20328*, 2025. 11, 12

[62] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv: 2104.09864*, 2021. 20

[63] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 12, 20

[64] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv: 2407.04620*, 2024. 2, 3

[65] Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context. *arXiv preprint arXiv: 2512.23675*, 2025. 5, 12

[66] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023. 11

[67] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Pannag R. Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *ROBOTICS*, 2024. doi: 10.48550/arXiv.2405.12213. URL https://arxiv.org/abs/2405.12213v2. 2, 3, 11, 12

[68] Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. *arXiv preprint arXiv: 2505.09561*, 2025. 11, 12

[69] Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv: 2603.03596*, 2026. 2, 11

[70] Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. *arXiv preprint arXiv: 2010.14876*, 2020. 11

[71] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NG7sS51zVF. 2

[72] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, Lars Liden, and Jianfeng Gao. Magma: A foundation model for multimodal ai agents, 2025. URL https://arxiv.org/abs/2502.13130. 12

[73] Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention. 22

[74] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2412.06464. 7

[75] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. 2, 11

[76] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv: 2603.16666*, 2026. 2, 11

[77] Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175. 12

[78] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv: 2505.23884*, 2025. 2, 3, 11, 12, 20

[79] Tianyu Zhao and Llion Jones. Fast-weight product key memory. *arXiv preprint arXiv: 2601.00671*, 2026. 12

[80] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *The Thirteenth International Conference on Learning Representations*, 2025. 11, 12

[81] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv: 2602.16710*, 2026. 7, 20

[82] Shaoting Zhu, Baijun Ye, Jiaxuan Wang, Jiakang Chen, Ziwen Zhuang, Linzhan Mou, Runhan Huang, and Hang Zhao. Ttt-parkour: Rapid test-time training for perceptive robot parkour. *arXiv preprint arXiv:2602.02331*, 2026. 12

[83] Christos Ziakas and Alessandra Russo. VITA: Zero-shot value functions via test-time adaptation of vision–language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=V35oo1SVGH. 12

A. Model Architecture, Training, and Deployment Details

A.1. Model Architecture

We instantiate RoboTTT on pretrained GR00T N1.7 (51), which consists of the Eagle model (59) as the vision-language model (VLM) backbone and a Diffusion Transformer (DiT) (52) as the action head. We add a TTT layer to each of its 16 DiT layers. The original DiT has 538M parameters; each TTT layer adds roughly 10M, for 690M in total. Each TTT layer contains a fast model, a two-layer MLP with GeLU activation (25), updated by standard gradient descent at test time; more sophisticated test-time optimizers such as Muon (33; 78) are left to future work. Following standard practice for TTT, we learn the inner test-time learning rate (63) on top of a constant base learning rate of 0.1. We use RoPE (62) for positional embeddings with $\theta_{\text{rope}} = 10000$.

A.2. Training

We pretrain on a mixture of tabletop bimanual robot data and egocentric human video data (81), curated to emphasize long trajectories; the trajectory length distribution is shown in Fig. A.1. We train all models on NVIDIA GB200 GPUs, using 16 GPUs per pretraining run and 8 per task-specific post-training run. Pretraining tunes only the newly added sequence-modeling layers (TTT or GDN) and freezes the other components of GR00T N1.7; post-training fine-tunes all parameters. During pretraining, we use a per-device batch size of 4 (global batch size 64) for context lengths of 4K and below, and 1 (global batch size 16) above. Post-training uses a 1K context length and a per-device batch size of 1. We pretrain for 30K steps and post-train for 20K steps. All models use the AdamW optimizer (48) with weight decay 1×10^{-5}. Pretraining uses the Warmup-Stable-Decay (WSD) schedule (28) with a peak learning rate of 2×10^{-5}; post-training uses a cosine schedule with a peak learning rate of 5×10^{-5}.

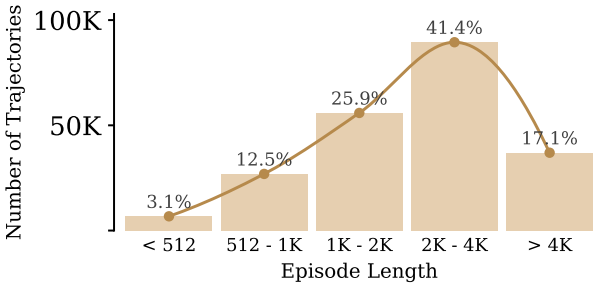

Figure A.1: **Trajectory length distribution of the pretraining mixture.**

A.3. Deployment

We deploy trained models on the YAM bimanual tabletop robot with four RealSense D405 cameras (top, bottom, left wrist, right wrist) streaming 480p RGB observations. Inference runs on a workstation with an NVIDIA RTX 5090 GPU, at a control frequency of 30 Hz.

B. Task Definition and Experiment Details

This section provides task definitions and experiment details.

B.1. Task Definition

All tasks are scored on a [0, 1] task completion scale via task-specific rubrics, detailed below.

Pup Go Car As shown in Fig. A.2, the robot assembles the yellow roof and the first wheel of the model car *"Pup Go Car"*. It picks up the roof, aligns the screw with the hole on the car body, and places the roof. It then grasps

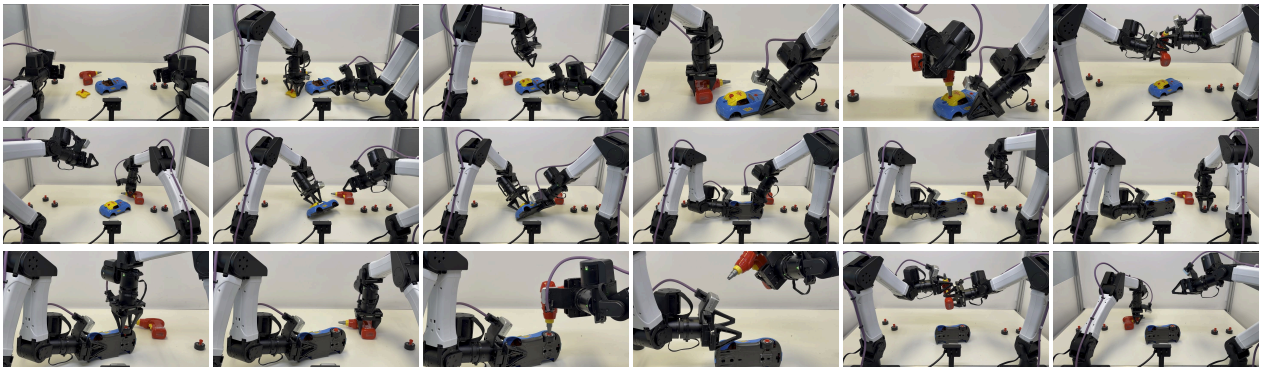

Figure A.2: **The "Pup Go Car" task.** The robot assembles the roof and first wheel of a model car across multiple stages, including screwing, drilling, bimanual handoffs, and a car flip.

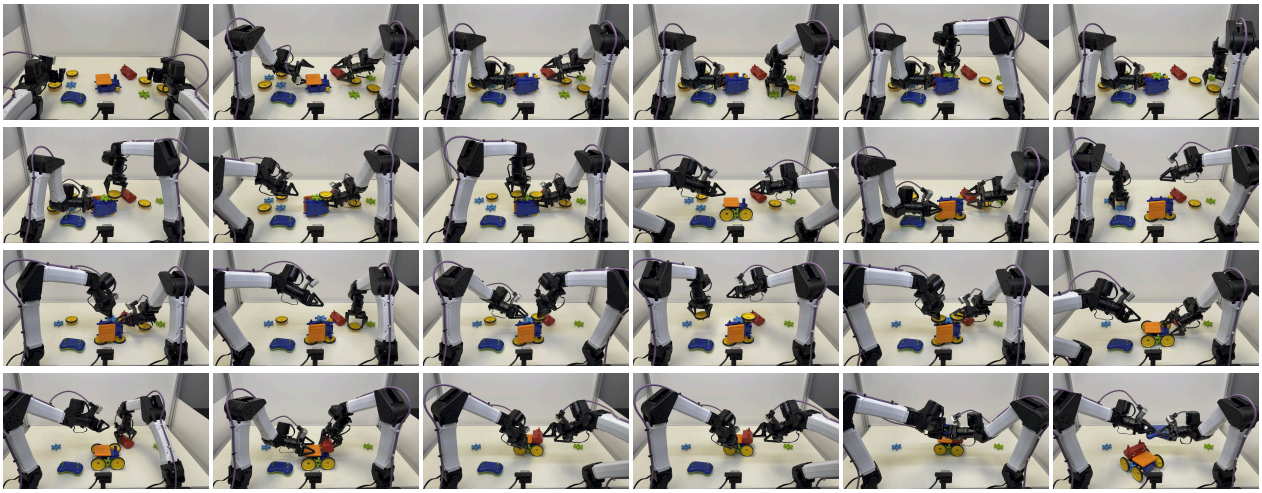

Figure A.3: **The "Gear Bot" task.** The robot installs gears and wheels on both sides of the chassis, flips it twice, attaches the robot head, and drives it with a remote.

the drill, aligns it with the roof screw, and tightens it, before handing the drill off to the right hand. The robot next flips the car with one hand and picks up a wheel with the right hand, which inserts the wheel into the car body. It then picks up the drill again and tightens the wheel, before finally handing the drill off to the left hand.

The scoring rubric is as follows: 0.05 if the roof is picked up; 0.1 if the roof is placed on the car body; 0.25 if the roof screw is properly inserted into the car body; 0.3 if the drill is picked up; 0.35 if the drill tip contacts the screw; 0.45 if the roof screw is fully tightened; 0.5 if the drill is handed to the right hand; 0.55 if the car body is flipped and stabilized; 0.6 if the tire is picked up; 0.75 if the tire is inserted into the car body; 0.8 if the drill is picked up; 0.85 if the drill is aligned with and contacting the wheel screw; 0.9 if the wheel is fully tightened; 1.0 if the drill is handed to the left hand. We allow at most two attempts for wheel assembly.

Gear Bot As shown in Fig. A.3, the robot assembles the entire toy model "Gear Bot". It installs one gear and two wheels on each side of the chassis, flipping the chassis twice so the shafts face upward. It then picks up the red "robot head" and inserts it onto the chassis. Finally, it picks up the remote control and pushes the joystick so the Gear Bot moves around. The scoring rubric is as follows: +0.1 for each chassis flip, +0.1 for each gear or wheel installation, +0.1 for "robot head" installation, and +0.1 if the remote control is used successfully.

Circuit As shown in Fig. A.4, the robot assembles circuit components on a board. Components include a red LED, green LED, colorful LED, lamp, motor, snap wire, press button, and switch. The robot assembles two or three pieces, following different assembly orders for the left and right components. Considering the component combinations and assembly orders, there are roughly 80 configurations. If the assembly includes a switch or press

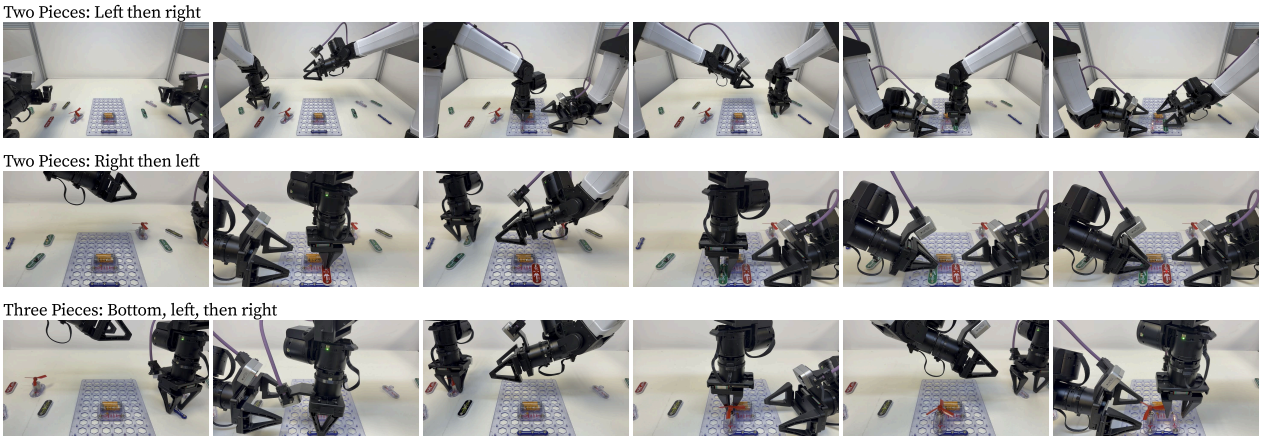

Figure A.4: **The "Circuit" task.** The robot assembles two or three circuit components on a board and powers the circuit on when a switch or button is present.

button, the robot must also turn it on after assembly. The scoring rubric is as follows. For two-piece assembly without a switch or press button, +0.5 per component installed. For two-piece assembly with a switch or press button, +0.33 per component installed and +0.33 if the circuit is turned on. For three-piece assembly without a switch or press button, +0.33 per component installed. For three-piece assembly with a switch or press button, +0.25 per component installed and +0.25 if the circuit is turned on. No partial credit is given if the assembly order is wrong.

B.2. Experiment Details

Baselines For the GR00T N1.7 and GR00T N1.7 Hist. baselines, we use the official implementation (51), extended to support history-frame input for GR00T N1.7 Hist.. For the GDN baseline, we replace each TTT layer with a Gated DeltaNet layer from the Flash Linear Attention library (73), keeping the layer placement, gating, and parameter count matched to RoboTTT.

Evaluation We deploy trained models on YAM bimanual tabletop robots. To ensure identical initial conditions across methods, we record the initial object placements for each task and reproduce them at evaluation time. We evaluate 20 rollouts for the Pup Go Car and Circuit tasks, and 10 rollouts for the Gear Bot task and for the Circuit task under the one-shot human-video setting, owing to their substantially longer evaluation time.