Extended Abstract

Robotic manipulation tasks that demand precision and multi-step coordination, such as object handovers or insertions, require policies that can reason over temporally extended sequences while remaining reactive to environmental changes. Action Chunking with Transformers (ACT) has demonstrated promise by issuing multi-step action chunks instead of per-timestep decisions, reducing compounding errors. However, its use of a fixed chunk size limits flexibility: long chunks hinder responsiveness during delicate motions, while short chunks are inefficient for stable phases. This motivates our work on adaptive chunking to improve robustness and efficiency across diverse task phases.

We propose an *Adaptive Action Chunk Selector*, a lightweight MLP module trained with PPO that dynamically selects the number of actions k' to execute from a precomputed ACT-predicted sequence. At each decision point, the selector receives visual observations, joint states, and a style embedding, and outputs a discrete chunk length from a fixed candidate set (e.g., 1, 30, ..., 100). The ACT policy remains frozen throughout, and only the selector is trained. The reward function guiding PPO training combines several terms to balance task success, control efficiency, and adaptability. A high-weight terminal reward reinforces successful task completion. A dense progress reward encourages continuous advancement by normalizing cumulative environment rewards. To promote efficiency, longer chunks are favored through a linear k'-scaled term. An exploration bonus incentivizes diversity by rewarding mid-range chunk sizes, and an adaptive shaping term encourages longer horizons during smooth progress and shorter ones when the agent needs tighter feedback. This structured reward enables phase-aware temporal adaptation: the agent learns to shorten its horizon during precise contact phases and extend it when performing predictable motions. All reward components were tuned empirically, and training was conducted across four ALOHA tasks (`sim_transfer_cube_scripted`, `sim_transfer_cube_human`, `sim_insertion_scripted`, `sim_insertion_human`) using on-policy rollouts. Careful shaping was critical to prevent degenerate behaviors, such as consistently selecting minimal horizons, and to ensure robust adaptation across varying task dynamics.

Empirical results show that the adaptive selector improves task success on three out of four benchmarks, with relative gains up to 20.8% on `sim_transfer_cube_human` tasks. These improvements are more pronounced in settings with higher variability and noise, where the ability to adjust control frequency provides a robustness advantage over fixed-horizon policies. On scripted tasks, where trajectories are more stable, performance remains comparable to vanilla ACT, confirming that adaptivity does not introduce regressions in well-structured scenarios. Qualitative rollouts reveal that the policy learns to modulate k' intelligently: shorter chunks are selected during contact-intensive phases requiring fine coordination, while longer chunks are used during free-space movements. This behavior aligns with human intuition and supports the goal of context-aware temporal abstraction. However, in the most challenging case—`sim_insertion_human`—adaptive chunking underperforms by 8%, primarily due to a reward shaping bias that overemphasizes longer horizons. This issue is compounded by distribution shift, as unused predictions from the frozen ACT model degrade feedback quality during short chunks.

These findings demonstrate that a lightweight, PPO-trained chunk-length selector can significantly enhance the robustness and flexibility of learned manipulation policies. By dynamically adjusting the temporal resolution of control, the system balances reactivity and efficiency, adapting to different task phases without sacrificing overall success. The results matter in practical robotics applications where sensing noise, actuation latency, and variable contact dynamics are common. Importantly, the approach maintains compatibility with frozen low-level policies, making it modular and easy to integrate. Nonetheless, the study highlights limitations in reward shaping sensitivity and the use of a fixed decoder horizon. Future work should explore joint training of the selector and decoder to mitigate distribution shift, and move toward task-level learning objectives that eliminate the need for hand-tuned shaping terms. Overall, this project offers a promising step toward more adaptable and reliable control in long-horizon, contact-rich robot manipulation tasks.

Adaptive action chunk selector

Ruopei Chen
Department of Mechanical Engineering
Stanford University
`rpchen@stanford.edu`

Ke Wang
Department of Mechanical Engineering
Stanford University
`kewalk@stanford.edu`

Yazhou Zhang
Department of Mechanical Engineering
Stanford University
`zhangyaz@stanford.edu`

Abstract

We present an adaptive action chunking framework that extends Action Chunking with Transformers (ACT) by introducing a PPO-trained module that dynamically selects the chunk length k' based on the current task context. This approach retains the temporal abstraction benefits of ACT while enabling finer control in contact-rich phases and efficient execution in stable segments. Evaluated on four ALOHA manipulation tasks, our method improves task success in three benchmarks—achieving up to 20.8% relative gains on `sim_transfer_cube_human` task—while matching baseline performance on well-structured scripted scenarios. These results demonstrate that adaptive chunking can enhance policy robustness and flexibility without retraining the underlying low-level policy.

1 Introduction

Executing fine-grained robotic manipulation tasks, such as bimanual handovers, requires not only accurate motor control but also the ability to manage temporal dependencies over multiple steps. A key challenge in imitation learning for such tasks lies in mitigating compounding errors and handling delays in feedback when predicting actions one step at a time.

The Action Chunking with Transformers (ACT) framework (Zhao et al., 2023) addresses this by generating multi-step action sequences, or action chunks, instead of issuing single low-level commands at each timestep. This approach reduces the frequency of decisions, leading to smoother trajectories, fewer compounding errors, and improved robustness. Importantly, by committing to a sequence of actions, the robot can execute longer-horizon behaviors without frequent re-evaluation, making it especially effective for real-world deployments with limited sensing and actuation fidelity.

However, while fixed-length chunking confers advantages in motion smoothness and decision stability, it imposes a rigid temporal structure on the policy. A fixed chunk size assumes that all task states are equally predictable and controllable, but this is not the case for dynamic and contact-rich manipulation scenarios. For instance, in phases requiring fine-grained coordination (e.g., inserting a connector or passing an object between grippers), a shorter chunk length allows more frequent observations and tighter control. Conversely, in coarse and stable movements (e.g., transporting an object through free space), longer chunks can improve efficiency without sacrificing precision. As a result, choosing the right chunk length is still an open question.

This project introduces an Adaptive Action Chunk Selector: a reinforcement learning (PPO) based policy module that dynamically adjusts the action chunk size according to the current task observation (Schulman et al., 2017). By integrating this adaptive mechanism into the ACT framework, we aim to retain the core benefits of action chunking while introducing the flexibility needed to respond to the environment in real time. This approach seeks to balance reactivity with efficiency, offering a more robust control policy for complex multi-phase robotic manipulation tasks.

Our contributions are:

- designed an adaptive action chunk selector that could instruct the agent the number of action to take according to the environment

2 Related Work

2.1 Action Chunking with Transformers

Zhao et al. (2023) proposed ACT for fine-grained bimanual manipulation. Instead of selecting one low-level command per timestep, ACT predicts a chunk of k actions in a single decision, shortening the effective planning horizon by a factor of k and curbing the compounding errors that hinder long tasks. The policy consists of a Transformer-based conditional variational autoencoder that models the variability in human demonstrations and generates multi-step, closed-loop behaviors. By "looking ahead" and committing to coherent action plans, ACT can make the precise, high-frequency adjustments required for tasks such as opening small containers or inserting batteries, where tiny deviations cause failure. ACT produces smoother and more coordinated motions than one-step behavioral cloning, which often falters on the precision and prolonged coordination demanded by delicate bimanual operations (Zhao et al., 2023). On the static ALOHA platform, ACT outperformed behavioral cloning and transformer baselines such as Behaviour Transformer (BeT) (Shafiullah et al., 2022) and RT-1 (Brohan et al., 2022) across six fine-manipulation tasks. Nevertheless, ACT's original implementation fixes the chunk length k for an entire episode, creating a responsiveness trade-off: a small k increases reactivity but adds computation and noise, whereas a large k yields smoother motions but slower feedback. The Mobile-ALOHA (Fu et al., 2024) extension couples ALOHA's bimanual arms with a mobile base for long-horizon tasks like cabinet handling and elevator use, yet it retains the same fixed chunk size.

A fixed chunk size assumes that all task states are equally predictable and controllable, but this is not the case for dynamic and contact-rich manipulation scenarios. Classical control offers a similar idea: adaptive-horizon MPC (Krener, 2018)lengthens its prediction window for straight-line motion and shortens it near sharp turns, maintaining real-time tracking without extra hardware. In robot learning, however, most hierarchical or chunk-based policies still operate with a fixed horizon, underscoring the need for an adaptive chunk-length mechanism.

This motivates us extending ACT with a mechanism for adaptive action chunking. In particular, enabling the chunk size to vary based on the current context can preserve ACT's strength in reducing compounding error while flexibly dialing the control granularity up or down as needed and, potentially, reducing the execution steps for improved efficency.

2.2 Proximal policy optimization (PPO) for adaptive control

PPO has become a popular algorithm for real-world robotics because it offers the combination of on-policy stability and implementation simplicity (Schulman et al., 2017; Tang et al., 2025). PPO's clipped-surrogate objective constrains each parameter update to stay within a small trust region, preventing the large gradient steps that can destabilize policies when an action choice (such as our chunk length k') has an outsized impact on future states. At the same time, the generalized-advantage estimator (GAE) and mini-batch updates yield good sample efficiency without the replay memory overhead of off-policy algorithms.

The algorithm's strengths have been demonstrated on hybrid-dynamics problems where discrete mode switches interact with continuous control. Discrete-time Hybrid Automata Learning (DHAL) uses a multi-critic PPO architecture and a beta-distribution policy to let a quadruped skateboard while automatically discovering and switching contact modes (Liu et al., 2025). The same stability that lets DHAL cope with abrupt mode transitions is crucial for our k-selector, which may occasionally jump from a long horizon to a small chunk without derailing the episode.

In conclusion, PPO provides the stability, discrete-action support, and data efficiency needed to train a reliable chunk-length controller without jeopardizing the frozen ACT low-level policy. Based on this, we decided to train the k'-selector by treating it as the actor for the PPO algorithm.

3 Method

3.1 Model architecture

We build on the ACT architecture and augment it with a trainable module to choose the chunk length k' at each decision point (Figure 1). The robot observation (images and joint states) and the style vector z are first processed by a frozen ACT model to propose a sequence of candidate actions and are simultaneously fed into a new policy network that decides how many of these actions to execute. The selected number of actions k' is then applied in the environment as one chunk, after which the cycle repeats with a new observation.

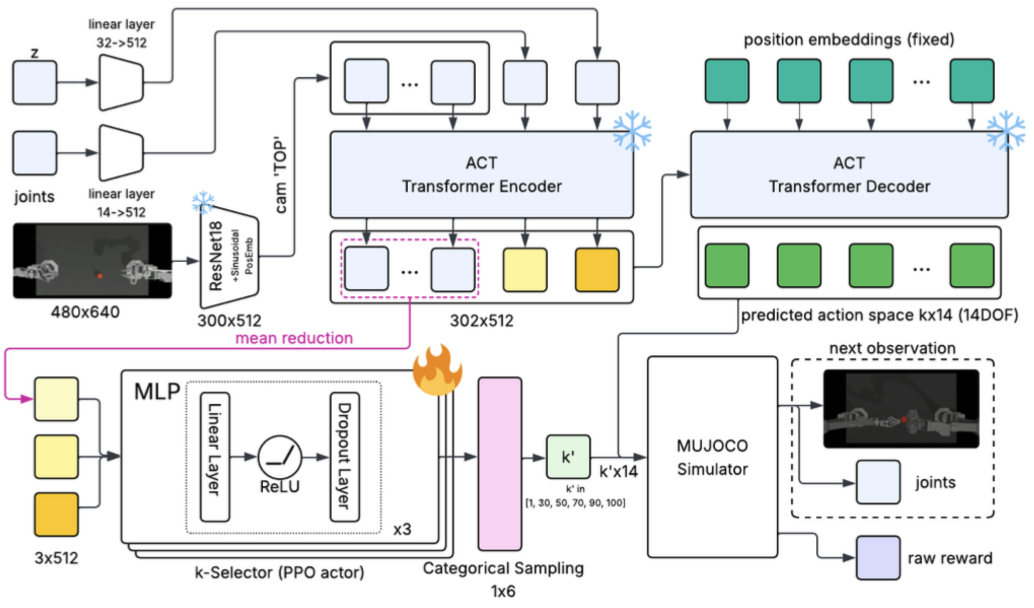

Figure 1: ACT with adaptive action chunk selector (i.e. k'-selector) architecture

Because PPO requires on-policy rollouts, all training is performed in the ALOHA MuJoCo simulator, which provides high-fidelity benchmarks for `sim_transfer_cube` and `sim_bimanual_insertion` tasks. These two tasks, available in both *scripted* and *human* demonstration variants, serve as the primary domains for training and evaluating our adaptive chunking policy.

The ACT decoder consists of three moduels: a ResNet image encoder, a transformer encoder, and a transformer decoder. The ResNet image encoder, followed by a linear layer, converts an input image of size 480×640 into a 300×512 embedding. Two additional linear layers project the joint-state vector ($14D$) and the style vector z ($32D$) into separate 512-D embeddings. The transformer encoder takes all 302 tokens and outputs 302 tokens with the same dimension (i.e. $512D$). These tokens are passed to the transformer decoder. An action-prediction head maps decoder output tokens to a low-level action of dimension $K \times 14$, where K is the predefined action chunk size in ACT and 14 is the DOF for ALOHA. ACT achieved the best task completion rate with $K = 100$.

The transformer encoder's 300 image token outputs are mean-pooled and concatenated with the joint-state and style embeddings, forming a single 1×1536 vector. Mean-pooling distills the tokens into an order-invariant global summary that preserves the overall visual context while guaranteeing a fixed-size input for the selector. This vector is passed as input to our lightweight k' selector. The k' selector is implemented as a three-block MLP, where each block consists of a linear projection, a ReLU activation, and a dropout layer. It outputs a discrete selection of k'. We parameterize k' as a categorical action: in the current implementation, the network chooses from a predefined set of possible chunk sizes 1, 30, 50, 70, 90, 100 representing various levels of granularity. These chunk sizes are selected empirically. At runtime, given the transformer's encoding of the latest observation, the k-selector outputs an index corresponding to a chunk length k'. We then execute the first k' actions out of the 100-length action sequence predicted by the ACT decoder in the MuJoCo simulator. The k'-selector is the actor of the PPO algorithm. The PPO's critic has the same architecture as the k'-selector for simplicity.

3.2 Reward design

The `sim_transfer_cube` task instructs the ALOHA to do the following: There is a small cube on the desk. The left gripper should pick up the cube and transfer to the right gripper. For each action taken, the simulated MuJoCo environment would give a reward R_{state_i} with respect to the status of the environment.

$$R_{\texttt{state\_i}} = \begin{cases} 1 & \text{if the right gripper touched the cube,} \\ 2 & \text{if the right gripper lifted the cube,} \\ 3 & \text{if the right gripper attempted transferring action,} \\ 4 & \text{if the left gripper hold the cube,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In the `sim_bimanual_insertion` task, the left and right arms need to pick up the socket and peg respectively from the table, and then insert in mid-air so the peg touches the "pins" inside the socket. The reward for this task is defined as

$$R_{\texttt{state\_i}} = \begin{cases} 1 & \text{if both grippers touched the objects,} \\ 2 & \text{if both grippers grasped the objects,} \\ 3 & \text{if peg and socket touching,} \\ 4 & \text{if successfully inserted,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Raw total reward is defined to be:

$$R_{\texttt{rawTotal}} = \sum_{i}^{k'} R_{\texttt{state\_i}} \tag{3}$$

Additional reward signals are required to provide rich intermediate feedback that distinguishes high-quality timing decisions from poor ones and pushes the policy toward the desired balance of reactivity, smoothness, and efficiency. As a result, we designed the reward function for the PPO to be:

$$R_{\texttt{final}} = R_{\texttt{success/fail}} + R_{\texttt{EpisodeProgress}} + R_{\texttt{ComputationEfficiency}} + R_{\texttt{exploration}} + R_{\texttt{EffectiveKSelection}} \tag{4}$$

where $R_{\texttt{success/fail}}$ is the reward signal for completing the task. The agent would get this reward only at the timestep it complete the task. This component has the highest weight because successful task completion is the primary learning objective. We found that a strong terminal reward also counter-balances the sparsity of success signals: the agent receives it only once, at the exact timestep when the goal is achieved, so the high weight prevents the policy from converging to behaviors that merely accumulate dense shaping rewards without actually solving the task.

$$R_{success/fail} = \begin{cases} 10.0, & \text{if } R_{state_i} == 4, \\ 0.0, & \text{otherwise.} \end{cases} \tag{5}$$

$R_{\texttt{EpisodeProgress}}$ is the reward for making meaningful progress toward task completion. If the agent makes good progress, the ratio $\frac{R_{\texttt{rawTotal}}}{\texttt{env\_max\_reward} \times 100}$ would be high. Dividing by $\texttt{env\_max\_reward} \times 100$ converts the raw cumulative reward into a fraction of the best-possible progress the agent could have achieved if every one of the 100 candidate actions had earned the maximum reward. We initially normalized by the current chunk length k' instead of the constant 100. That choice unintentionally inflated the progress reward for short chunks: the smaller the k', the smaller the denominator, so the ratio, and thus the reward, grew larger. The selector quickly learned to exploit this loophole, collapsing to k'=1 after only a few training epochs.

$$R_{\texttt{EpisodeProgress}} = 3.0 \times \min\left(\frac{R_{rawTotal}}{\texttt{env\_max\_reward (i.e. 4)} \times 100}, 1.0\right) \tag{6}$$

and $R_{\texttt{ComputationEfficiency}}$ is added to avoid unnecessary short action chunk, because our argument is that the agent should only take small action chunk when the environment is more stochastic or uncertain. Conceptually, selecting a short chunk forces the agent to "pay" more attention—querying sensors and the ACT model more often. By making the reward proportional to k', we impose a price for this extra attention, mirroring ideas in adaptive-computation-time networks where each additional pondering step incurs a penalty. The reward would be higher if k' is closer to 100 (k value for ACT).

$$R_{\texttt{ComputationEfficiency}} = \frac{k'}{100} \tag{7}$$

and $R_{\texttt{exploration}}$ term explicitly promotes diversity in chunk-length choices. This assigns the highest bonus to mid-range values (e.g., k' = 50 or 70). Without this explicit diversity incentive, the selector gravitates toward whichever extreme gives the easiest immediate reward, either the tiny horizon favoured by the EpisodeProgress term or the maximal horizon rewarded by ComputationEfficiency. Once this bonus was introduced, the k'-selector began sampling intermediate horizons more frequently, resulting in a mixture of short, medium, and long action chunks.

$$R_{\texttt{exploration}} = 0.5 \times \max(\{1 - 2|\frac{k'-1}{100} - 0.7|\}, 0.1) \tag{8}$$

and $R_{\texttt{EffectiveKSelection}}$ signal ties each chunk-length decision directly to the ongoing episode-progress metric. Whenever the agent is advancing smoothly, reflected by a steadily increasing progress score, it earns an additional reward for selecting larger k' values, thereby encouraging longer and more efficient open-loop executions. Conversely, if progress stalls or deteriorates, the reward function instead favors smaller k' values, prompting the policy to shorten its horizon, obtain a fresh observation sooner, and regain control.

$$R_{\texttt{EffectiveKSelection}} = \begin{cases} \frac{k_j - k_{\max}}{k_{\max} - k_{\min}}, & \text{if } \frac{R_{\texttt{rawTotal}}}{\texttt{env\_max\_reward} \times 100} > 0.7, \\ 1 - \frac{k_j - k_{\max}}{k_{\max} - k_{\min}}, & \text{otherwise.} \end{cases} \tag{9}$$

We emphasize that the specific weights and auxiliary components of the reward function were chosen through empirical trial-and-error rather than systematic optimization. Consequently, the present shaping scheme should be regarded as a pragmatic baseline, not as a optimal design.

4 Experimental Setup

We select the following tasks: 1) $sim\_transfer\_cube\_scripted$, 2) $sim\_transfer\_cube\_human$, 3) $sim\_bimanual\_insertion\_scripted$, and 4) $sim\_bimanual\_insertion\_human$. Figure 2 shows the task definitions for each task. The raw environment reward for each task is explained in Section 3.2.

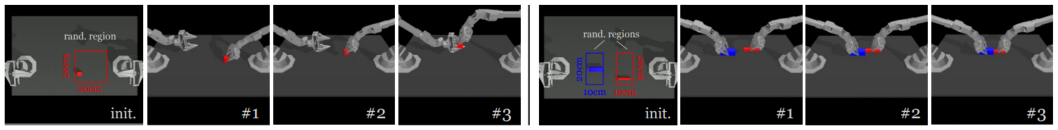

Figure 2: **Simulated Task Definitions. Left – Cube Transfer:** Transfer the red cube to the other arm. The right arm touches (#1) and grasps (#2) the red cube, then hands it to the left arm. **Right – Bimanual Insertion:** Insert the red peg into the blue socket. Both arms grasp (#1), let socket and peg make contact (#2), and complete the insertion. (Zhao et al., 2023)

For every benchmark task we first trained the ACT backbone with identical hyperparameters following previous publication (Zhao et al., 2023). The action chunk was fixed at $k = 100$ (ACT has highest task success rate at $k = 100$). After this stage the ACT weights were frozen, and they are used as our baseline models. Subsequently, we trained the k'-selector for 200 epochs, collecting 50 simulated episodes per epoch for on-policy updates. Each episode has a `max_timestep` of 400. The k'-selector is the actor for the PPO algorithm, and PPO's critic has same architecture as k'-selector. In total, PPO model has around 3M parameters, and training took about two hours for each task.

During evaluation we compare ACT's task-completion rate with that of ACT with adaptive action chunk selector. To further assess the k'-selector's behaviour, we record the chosen k' at every decision point throughout each evaluation episode.

5 Results

5.1 Quantitative Analysis

We compare the success rate for vanilla ACT and ACT with k' selector (Figure 3). Both agents reach the same high success rate (92%) in `sim_transfer_cube_scripted` indicating that the vanilla ACT policy is already nearly optimal for this well-structured scenario. Adaptive chunking offers no measurable benefit but also incurs no penalty. When evaluated on demos collected via teleoperation (`sim_transfer_cube_human`), the adaptive selector raises success from 48% to 58%. A 20.8% relative improvement suggests that adaptivity helps compensate for the greater variability and noise typical of human trajectories. In `sim_insertion_scripted`, success rate improves from 48% to 54%. These improvements show that introducing k' selector can indeed make the policy more robust. However, for the task `sim_insertion_human`, the performence for ACT with adaptive action chunk selector is 8% lower than that for vanilla ACT. This may because the vanilla ACT has low performance (26%), causing the k' selector to inherit a noisy value landscape.

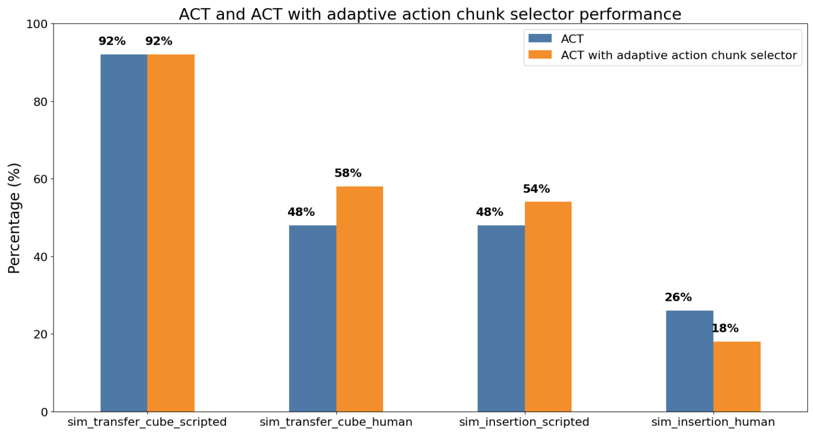

Figure 3: Task success rate comparison between vanilla ACT and ACT with adaptive action chunk selector for four different tasks

5.2 Qualitative Analysis

To further analyze the effect of k' selector, we show the variation in k' values per rollout (i.e. episode) for 50 episodes (Figure 4 and 5). Chunk Number means the number of decision made. The plot is intepreted as during the i^{th} (y-axis) episode the j^{th} (x-axis) decision made is to take k' actions. For both tasks, we could find a phase-dependent pattern in the horizon selection. The agent consistently selects smaller action chunk sizes (lower k' values) during phases where the two robot arms must coordinate closely. This behavior is expected, as reducing k' allows the agent to observe the environment more frequently, increasing safety and precision at critical interaction points. Qualitative video roll-outs further highlight the benefit of adaptive chunking. In the vanilla ACT baseline, if the robot misses its grasp on the cube it blindly executes the remainder of the 100-step chunk, wasting time while the object lies untouched. With the adaptive selector, however, the robot commits only to a short horizon during the initial contact phase; when the grasp fails, it recognises the error at early stage and replans the trajectory. This ability to detect mistakes early translates into smoother recoveries and fewer irrecoverable failures in the human-demonstration tasks.

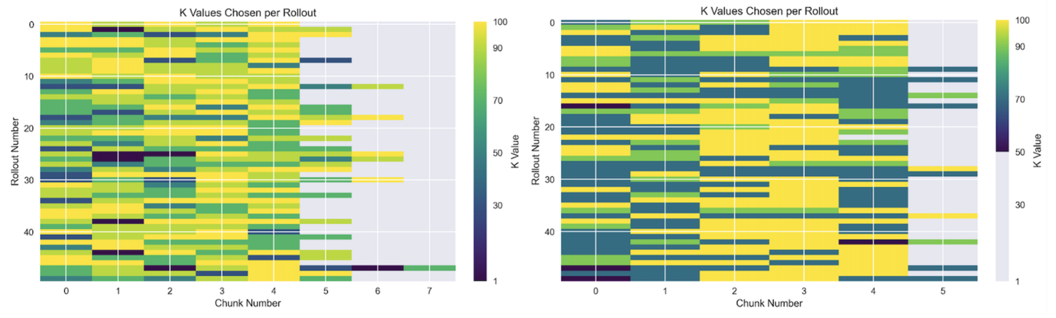

Figure 4: Variation in k' selection along single episode for 50 rollouts. LEFT: `sim_transfer_cube_scripted` RIGHT:`sim_transfer_cube_human`

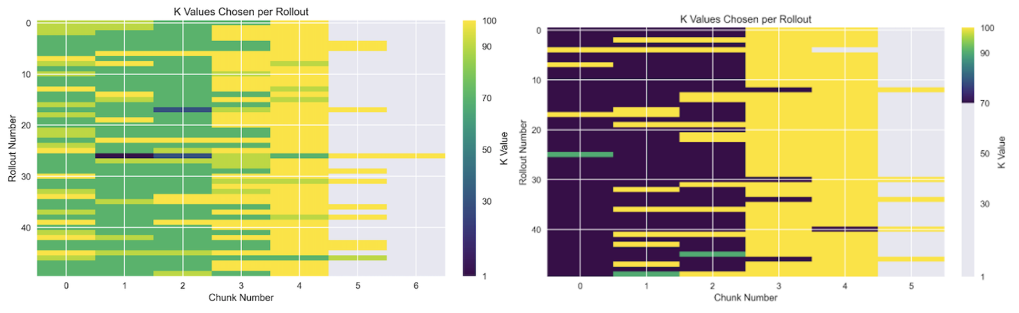

Figure 5: Variation in k' selection along single episode for 50 rollouts. LEFT: `sim_insertion_scripted` RIGHT:`sim_insertion_human`

A closer inspection of horizon statistics on the human-demo tasks reveals that the selector effectively abandoned short horizons: during evaluation it never chose $k' < 70$ for `sim_transfer_cube_human` and never went below $k' = 50$ for `sim_insertion_human`. Training logs confirm that the policy experimented with small chunks only in the first ten epochs before drifting toward consistently large values. This collapse is largely attributable to reward-design bias—our Computation-Efficiency term grows linearly with k' while the Episode-Progress bonus is normalised by a constant 100, so on noisy human demonstrations long chunks accumulate higher expected returns. Because executing very short horizons exposes the robot to states the frozen ACT decoder never saw, early rollouts with $k' < 50$ often failed, further suppressing their value estimates. The exploration bonus, tuned on smoother scripted data, was too weak to counteract these penalties, and PPO's clipped updates then locked probability mass onto the safer large-chunk actions. Together these effects explain why the adaptive policy stopped selecting small k' on human data and, as a consequence, failed to outperform vanilla ACT in the more challenging insertion scenario.

6 Discussion

Our experiments demonstrate that layering a lightweight, PPO-trained k' selector on top of a frozen ACT backbone can boost task completion rates in three of the four benchmark settings, especially when demonstrations are noisy or the task naturally alternates between coarse free-space motion and fine contact phases. The selector extends the action horizon during predictable transit segments and shortens it during delicate interactions, enabling earlier error detection and smoother recoveries than vanilla ACT. Such context-aware temporal adaptation could make learned policies more reliable on low-cost robots, advancing the practicality of learning-based manipulation in real homes and factories.

However, several limitations and open challenges remain. First, the system inherits a fixed 100-step decoder horizon from the vanilla ACT training. When the selector truncates a chunk after only a few steps, the decoder's later predictions go unused, creating a distribution shift that can hurt performance in highly variable scenarios, as observed in `sim_insertion_human`. Second, the selector currently chooses from a coarse, hand-crafted set of six chunk lengths. Allowing a finer horizon space may yield larger gains. Third, our reward shaping was tuned on scripted data by trial-and-error, and the same weights biased the selector toward long chunks on human demonstrations—highlighting the sensitivity of adaptive-horizon learning to reward scales. Indeed, designing and balancing the multiple reward terms proved the most time-consuming aspect of the project. A promising next step is to train the k' selector end-to-end with the ACT decoder, replacing manual reward engineering with a single, task-level objective and thereby mitigating the brittleness introduced by hand-tuned shaping weights.

7 Conclusion

This work shows that a lightweight module, trained with PPO and placed on top of a frozen ACT policy, can endow an imitation-learned manipulator with context-aware temporal adaptation. By letting the robot lengthen or shorten its action horizon on the fly, we bridge the gap between the smoothness of large, fixed action chunks and the reactivity of step-wise control, without retraining the underlying transformer. Remarkably, even a modest six-way categorical selector boosts robustness when supported by well-balanced rewards and adequate exploration. Such adaptive-horizon control offers a low-cost route to more dependable manipulation on resource-constrained hardware, where frequent replanning is expensive yet occasional fine-grained corrections remain crucial.

Future work includes (i) retrain ACT with a longer decoder horizon and a finer k' space to give the selector greater latitude, (ii) design multi-phase manipulation benchmarks that better stress-test adaptive chunking, and (iii) measure not only success rates but also the timesteps required to complete tasks, comparing vanilla ACT against our adaptive variant to quantify efficiency gains.

8 Team Contributions

- **Ruopei Chen** ACT and k' selector integration, PPO reward design, report
- **Ke Wang** PPO pipeline design, reward design, report
- **Yazhou Zhang** ACT training, PPO pipeline design, report

References

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817* (2022).

Zipeng Fu, Tony Z Zhao, and Chelsea Finn. 2024. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117* (2024).

Arthur J Krener. 2018. Adaptive horizon model predictive control. *IFAC-PapersOnLine* 51, 13 (2018), 31–36.

Hang Liu, Sangli Teng, Ben Liu, Wei Zhang, and Maani Ghaffari. 2025. Discrete-time hybrid automata learning: Legged locomotion meets skateboarding. *arXiv preprint arXiv:2503.01842* (2025).

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

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. 2022. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems* 35 (2022), 22955–22968.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. 2025. Deep reinforcement learning for robotics: A survey of real-world successes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 28694–28698.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. 2023. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705* (2023).