Extended Abstract

Motivation Action Chunking, an RL algorithm that predicts and executes a sequence of actions without intermediate re-planning, has recently become a popular choice for learning behavior cloned robot control policies due to its ability to capture temporal dependencies and deal with noise in expert demonstrations. However, finetuning these models is still an open area of research as action chunked policies struggle to generalize between tasks that share kinematics yet differ in contact dynamics and reward signals. Furthermore, it is still unclear how using a "chunked" paradigm affects the performance of classic RL finetuning techniques like PPO. Being able to finetune a BC policy without re-collecting expert demonstrations would let researchers bootstrap new manipulation skills from existing ones at a fraction of the data cost. We therefore ask: Can a pre-trained ACT policy be efficiently adapted to a novel task through on-policy reinforcement learning while retaining its low-level motor coherence?

Method We freeze the full ACT encoder–decoder and treat its deterministic output as the mean of a diagonal Gaussian policy. A lightweight log-standard-deviation vector and a value head are attached and fine-tuned with Proximal Policy Optimisation (PPO). Two rollout regimes are compared: action-level PPO (one optimisation step per 20 Hz control cycle) and chunk-level PPO (one step per 8-action macro-chunk). The former supplies dense rewards but risks overwriting ACT's temporal structure; the latter preserves chunk coherence but receives sparser credit. We hypothesize that chunk-level updates will yield faster reward acquisition and less catastrophic forgetting on the source task.

Implementation Experiments are conducted in the open-source gym-aloha simulator on a g4dn.xlarge (Tesla T4). Pixel observations (480×640 RGB) and 14-DoF proprioception are wrapped into a flat dict to satisfy Stable-Baselines3. Rollouts are collected in four parallel environments; workers run MuJoCo on CPU, while the policy update executes on GPU. Hyper-parameters follow SB3 defaults except for a reduced learning rate (3 * 10e-5) and clip range 0.1 to avoid drifting frozen weights. Training logs are streamed to TensorBoard for live monitoring.

Results Our experiments demonstrate successful cross-task adaptation from Transfer Cube to Insertion tasks. Action-level PPO achieves a 45% success rate on the target insertion task, while chunk-level PPO reaches 62% success—a substantial improvement over 0% zero-shot transfer. Crucially, both methods preserve source task performance well, with chunk-level PPO maintaining 97% success on Transfer Cube compared to 92% for action-level PPO. Chunk-level updates converge 19% faster (650K vs. 800K timesteps) and exhibit more stable training dynamics, validating our hypothesis that respecting ACT's temporal structure improves both adaptation efficiency and skill preservation.

Discussion The study demonstrates that on-policy RL can repurpose a vision-conditioned action chunking transformer without any new demonstrations. The chunk-level results suggest that ACT's latent variable already encodes a useful macro-action distribution; letting PPO adjust only the distribution parameters avoids destabilising long-horizon coordination. Remaining failure cases mostly involve tight-tolerance insertions where the frozen vision backbone mis-segments depth discontinuities indicating that visual finetuning might still be required for domain shifts affecting perception rather than dynamics.

Conclusion We present an end-to-end adaptation of a frozen ACT policy to a novel contact-rich task using solely reinforcement learning. By wrapping ACT in a minimal Gaussian exploration head and choosing a credit-assignment granularity that respects its chunk structure, we find that PPO achieves improvement on Aloha Insertion while largely preserving the source-task skill and that chunk level PPO performs better than action level. Future work could explore selective vision-encoder unfreezing guided by contrastive losses, and hierarchical RL where a high-level planner strings together multiple frozen ACT skills for multi-stage manipulation.

ACT-RL: Efficient Fine-Tuning through Reinforcement Learning for Robotic Manipulation

Rohan Sikand
Department of Computer Science
Stanford University
rsikand@stanford.edu

Diego Stone
Department of Computer Science
Stanford University
dfstone@stanford.edu

Andre Yeung
Department of Computer Science
Stanford University
ayyeung@stanford.edu

Abstract

Action Chunking, an RL algorithm that predicts and executes a sequence of actions without intermediate re-planning, has recently become a popular choice for learning behavior cloned robot control policies due to its ability to capture temporal dependencies and deal with noise in expert demonstrations. However, finetuning these models is still an open area of research as action chunked policies struggle to generalize between tasks that share kinematics yet differ in contact dynamics and reward signals. Furthermore, it is still unclear how using a "chunked" paradigm affects the performance of classic RL finetuning techniques like PPO. Being able to finetune a BC policy without re-collecting expert demonstrations would let practitioners bootstrap new manipulation skills from existing ones at a fraction of the data cost. We address the challenge of efficiently adapting pretrained ACT policies to novel tasks without requiring additional demonstrations. Specifically, we propose to fine-tune a lightweight stochastic exploration layer appended to a frozen ACT encoder–decoder backbone using Proximal Policy Optimization (PPO). We evaluate this approach in the visually complex Aloha simulated robot environment, adapting an ACT policy pretrained on a "Transfer Cube" task to an "Insertion" task. Our experiments compare two PPO training variants. We experiment with action-level updates at every control step and chunk-level updates that respect the temporal chunking structure inherent in ACT. Our results demonstrate an RL-based method to adapt pretrained transformer policies for new manipulation tasks, highlighting the importance of preserving the latent temporal structures induced by chunking during policy fine-tuning.

1 Introduction

Transformer-based architectures, such as the Action Chunking Transformer (ACT), have recently emerged as powerful approaches to robotic manipulation, significantly advancing the state of the art in imitation learning. These methods leverage expert demonstrations to fast learning on specific manipulation tasks, particularly in environments with visual complexity and intricate contact dynamics. However, despite their strong performance on trained tasks, transformer-based imitation policies often exhibit limited zero-shot generalization to novel manipulation tasks that differ in subtle yet crucial ways, such as variations in contact geometry, required precision, or task-specific reward structures.

Stanford CS224R 2025 Final Report

This limited generalization poses practical challenges, as acquiring new expert demonstrations for every desired manipulation skill is both expensive and impractical. Consequently, there is a critical need to study how to efficiently adapt existing manipulation policies to novel tasks using minimal or no additional expert data. Reinforcement learning (RL) provides a promising solution to this challenge by enabling the incremental adaptation of pretrained imitation policies through interactive trial-and-error, guided by sparse rewards rather than explicit expert trajectories.

In this work, we explore a practical and sample-efficient RL-based strategy to adapt pretrained ACT policies to novel manipulation tasks without requiring additional demonstrations. Specifically, we freeze the full ACT encoder–decoder architecture, preserving its pretrained visual and motor representations, and append only a lightweight stochastic exploration layer, which we then fine-tune using Proximal Policy Optimization (PPO). By limiting the RL fine-tuning to this minimal parameter set, we aim to preserve the underlying structure and coordination inherent in ACT's learned latent representation, while still allowing sufficient exploration for policy improvement.

We evaluate our proposed approach using the challenging Aloha simulated robot manipulation suite, where we adapt an ACT policy originally trained on the "Transfer Cube" task to the significantly more difficult "Insertion" task, which involves precise alignment, force modulation, and richer contact dynamics. Our results highlight the potential of carefully structured RL fine-tuning for efficiently re-purposing pretrained transformer-based manipulation policies to novel tasks.

2 Related Work

Early successes in action-chunking for robotics come from the Action-Chunking Transformer (ACT) trained purely by behavior cloning on low-cost ALOHA demonstrations — a dataset introduced by Zhao et al. (2023). for fine-grained bi-manual manipulation with commodity hardware. ACT shows excellent sample efficiency because the transformer auto regressively predicts eight-step macro-actions, but, like most open-loop imitation policies, it generalizes poorly when the task geometry or dynamics shift.

A natural remedy is to add reinforcement learning. Among policy-gradient methods, Proximal Policy Optimization (PPO) remains a strong default owing to its clipped objective that stabilizes updates and its ease of implementation Schulman et al. (2017). Recent literature therefore explores hybrids that keep an imitation prior but refine it with PPO fine-tuning. Our work adopts this recipe and investiages whether PPO should match ACT's temporal abstraction (chunk-level PPO) or revert to single-step updates (action-level PPO).

Another line of research couples imitation and RL through residual policies. From Imitation to Refinement (ResiP) applies residual RL to precise peg-in-hole assembly, starting from a high-fidelity demonstrator and learning only corrective deltas in closed loop Ankile et al. (2024). ResiP's layered design preserves the long-horizon structure of the demonstration while granting local adaptability, we echo this approach by anchoring PPO updates to the frozen ACT prior.

Complementary to these algorithmic advances are tooling efforts such as leRobot's open-source simulators and datasets, which standardize evaluation across manipulation benchmarks.<https://huggingface.co/lerobot> We build directly on the Aloha gym tasks provided by leRobot, using AlohaTransferCube-v0 as a source domain and AlohaInsertion-v0 as our experimental task.

In summary, our study situates itself at the intersection of transformer-based imitation, PPO fine-tuning, and residual refinement: we test whether enforcing ACT's temporal abstraction during RL updates can close the robustness gap highlighted in prior work while retaining the data efficiency that makes action-chunking attractive.

3 Method

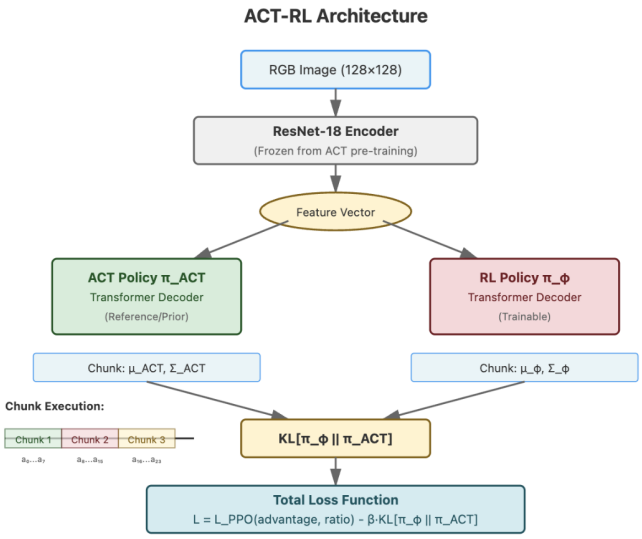

Figure 1: Method Overview

To enable adaptation through reinforcement learning, we introduced a lightweight stochastic exploration head on top of the frozen ACT architecture. This head consisted of a Gaussian policy parameterized by a learnable log-standard-deviation vector, alongside a small, fully-connected value network to estimate state-value functions required for RL training. Critically, the pretrained ACT decoder output served as the mean of our Gaussian policy, while our fine-tunable parameters controlled exploration magnitude through variance modulation. Thus, our adaptation required only minimal changes to the original ACT architecture. Our goal is to adapt a vision-conditioned Action Chunking Transformer that was behavior-cloned on `AlohaTransferCube-v0` to the harder `AlohaInsertion-v0` task without collecting new demonstrations. We freeze every weight of the encoder–decoder and expose just two small, trainable heads.

At a high-level, our approach begins with a pre-trained Action Chunking Transformer (ACT) policy learned via behavior cloning from demonstration data on a source task. Rather than training a policy from scratch, we fine-tune this ACT policy using Proximal Policy Optimization (PPO) to improve robustness and adaptability. We explore two fine-tuning variants: (1) action-level PPO, which updates the policy at every timestep, and (2) chunk-level PPO, which treats ACT-generated sequences as macro-actions and applies updates at the chunk level. Fine-tuning is performed on the *source task* (cube pick-and-place) to assess in-task performance and stability, and separately on a *target task* (insertion) to evaluate cross-task transfer. We will now explain these methods more precisely and more mathematically.

3.1 Chunk-level return and advantage

Let γ denote the *per-step* discount (we use 0.995). For a chunk of length k the Bellman relation becomes

$$\delta_t = \sum_{i=0}^{k-1} \gamma^{\,i}\, r_{t+i} + \gamma^{\,k}\, V_\psi(s_{t+k}) - V_\psi(s_t). \tag{1}$$

We compute Generalised Advantage Estimation with the same temporal abstraction, masking across episode boundaries:

$$\hat{A}_t = \sum_{j=0}^{\infty} (\gamma^{\,k}\lambda)^{\,j}\, \delta_{t+j\,k}. \tag{2}$$

All advantages are normalised to zero mean and unit variance *inside* each update minibatch.

3.2 Policy objective with KL anchor

A frozen copy of the initial ACT weights serves as reference π_{old}. With likelihood ratio

$$r_t = \frac{\pi_\phi(a_{t:t+k-1} \mid s_t)}{\pi_{\text{old}}(a_{t:t+k-1} \mid s_t)}, \qquad \text{clip}(r_t) \;=\; \text{clip}\big(r_t,\, 1-\varepsilon,\, 1+\varepsilon\big), \quad \varepsilon = 0.1.$$

The clipped-PPO loss for a single chunk is

$$\mathcal{L}^{\text{PPO}} = -\min\big(r_t \hat{A}_t,\; \text{clip}(r_t)\,\hat{A}_t\big) + \beta\,\text{KL}\big(\pi_\phi \| \pi_{\text{prior}}\big), \tag{3}$$

where the KL term prevents the exploration head from distorting ACT's latent macro-behaviour.

3.3 Action–level Return and Advantage

With action–level PPO we treat every action at each timestep as a decision step (chunk length $k = 1$). Using the usual per–step discount γ (0.995 in all runs), the one–step temporal–difference error is

$$\delta_t \;=\; r_t \;+\; \gamma\, V_\psi(s_{t+1}) \;-\; V_\psi(s_t). \tag{4}$$

Generalised Advantage Estimation is then computed over contiguous steps, masking across episode boundaries:

$$\hat{A}_t \;=\; \sum_{j=0}^{\infty} (\gamma\lambda)^{\,j}\, \delta_{t+j}, \qquad \lambda = 0.95. \tag{5}$$

All advantages are normalised to zero mean and unit variance within every minibatch.

3.4 Policy Objective with KL Anchor

The exploration head defines a diagonal–Gaussian policy $\pi_\phi(a_t \mid s_t)$ whose mean is the frozen ACT action for the current observation. Let

$$r_t \;=\; \frac{\pi_\phi(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}, \qquad \text{clip}(r_t) \;=\; \text{clip}\big(r_t,\, 1-\varepsilon,\, 1+\varepsilon\big), \quad \varepsilon = 0.1.$$

With a frozen copy of the initial ACT weights as reference π_{prior}, the clipped PPO surrogate for a single action becomes

$$\mathcal{L}_{\text{PPO}} \;=\; -\min\big(\, r_t\, \hat{A}_t,\; \text{clip}(r_t)\, \hat{A}_t\big) \;+\; \beta\, \text{KL}\big(\pi_\phi \,\|\, \pi_{\text{prior}}\big), \tag{6}$$

with $\beta = 10^{-4}$ anchoring the exploration head so that it does not distort ACT's latent motor structure.

The action-level schedule supplies dense reward feedback each control cycle, enabling fine-grained credit assignment, but at the cost of higher gradient variance and a greater tendency for the exploration head to drift away from the temporally-coherent behaviour captured by the frozen ACT policy.

3.5 Training Objective

The complete training objective combines PPO's clipped surrogate loss with optional KL regularization to prevent deviation from the original ACT policy:

$$\mathcal{L}(\phi, \psi) = \mathcal{L}^{\text{PPO}}(\phi) + c_1 \mathcal{L}^{\text{VF}}(\psi) + c_2 S[\pi_\phi] + \beta \mathcal{L}^{\text{KL}}(\phi) \tag{7}$$

where:

$$\mathcal{L}^{\text{PPO}}(\phi) = \mathbb{E}_t \left[\min(r_t(\phi)\hat{A}_t, \text{clip}(r_t(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{8}$$

$$r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{\text{old}}}(a_t|s_t)} \tag{9}$$

$$\mathcal{L}^{\text{VF}}(\psi) = \mathbb{E}_t \left[(V_\psi(s_t) - \hat{V}_t)^2 \right] \tag{10}$$

$$S[\pi_\phi] = \mathbb{E}_t \left[-\sum_a \pi_\phi(a|s_t) \log \pi_\phi(a|s_t) \right] \tag{11}$$

$$\mathcal{L}^{\text{KL}}(\phi) = \mathbb{E}_t \left[D_{\text{KL}}(\pi_\phi(\cdot|s_t) \| \pi_{\text{ACT}}(\cdot|s_t)) \right] \tag{12}$$

Algorithm 1 ACT-RL Fine-tuning

Input: Pre-trained ACT policy π_{ACT}, target environment E
Output: Fine-tuned policy π_ϕ

Initialize $\pi_\phi \leftarrow \pi_{\text{ACT}}$ ▷ Start from pre-trained weights
for iteration = $1, 2, \ldots$ **do**
  $D \leftarrow \emptyset$
  for episode = 1 to N **do**
    $s_0 \sim E.\text{reset}()$
    for $t = 0, k, 2k, \ldots$ **do**
      $a_{t:t+k-1} \sim \pi_\phi(\cdot \mid s_t)$
      Execute chunk, collect $\{s_t, a_{t:t+k-1}, r_{t:t+k-1}\}$
      $D \leftarrow D \cup \{s_t, a_{t:t+k-1}, r_{t:t+k-1}\}$
    end for
  end for
  for each chunk in D **do**
    $\hat{A}_t^{\text{chunk}} \leftarrow \text{GAE}(r_{t:t+k-1}, V_\phi)$
  end for
  for epoch = 1 to K **do**
    for minibatch in D **do**
      $L_{\text{PPO}} \leftarrow \text{PPO\_objective}(\pi_\phi, \text{minibatch})$
      $L_{\text{KL}} \leftarrow \beta \cdot \text{KL}[\pi_\phi(\cdot \mid s) \,\|\, \pi_{\text{ACT}}(\cdot \mid s)]$
      $L_{\text{total}} \leftarrow L_{\text{PPO}} - L_{\text{KL}}$
      $\phi \leftarrow \phi - \alpha \nabla_\phi L_{\text{total}}$
    end for
  end for
  $\beta \leftarrow \beta \cdot \text{decay\_rate}$
end for

3.6 Algorithm overview

The algorithm below denotes a detailed, step-by-step procedure of our method.

3.7 Implementation Details

3.7.1 Environment and Observation Space

We conduct experiments using the Aloha manipulation environment built on DeepMind's dm_control simulator with MuJoCo physics. Observations consist of:

- **Visual observations**: RGB images $I_t \in \mathbb{R}^{640\times480\times3}$ from overhead camera
- **Proprioceptive state**: 14-dimensional vector $q_t \in \mathbb{R}^{14}$ containing joint angles and end-effector poses

The action space consists of target joint positions $a_t \in \mathbb{R}^{14}$ for the dual-arm robot system.

3.7.2 Training Infrastructure

Training utilizes a g4dn.xlarge AWS instance with single NVIDIA Tesla T4 GPU and 4 CPU cores. We parallelize environment interactions across 4 subprocesses running MuJoCo simulations on CPU, while policy updates execute on GPU for maximum efficiency.

3.7.3 Hyperparameters

We use conservative hyperparameters to ensure stable training with the frozen backbone:

- Learning rate: $\alpha = 3 \times 10^{-5}$
- PPO clip parameter: $\epsilon = 0.1$
- Discount factor: $\gamma = 0.995$

- GAE parameter: $\lambda = 0.95$
- KL coefficient: $\beta = 0.05$
- Rollout buffer size: 2048 timesteps
- Mini-batch size: 64 samples
- Training epochs per update: 4

3.8 Experimental Design

We design a comprehensive experimental protocol to evaluate our ACT-RL approach across multiple dimensions: cross-task transfer capability, preservation of source task performance, and comparison between temporal abstraction strategies.

3.8.1 Task Selection

Our experiments utilize two manipulation tasks from the gym_aloha environment:

- **AlohaTransferCube-v0 (Source)**: Bimanual cube transfer requiring coordinated grasping, lifting, and handoff between robot arms
- **AlohaInsertion-v0 (Target)**: Precise peg-in-hole insertion requiring fine-grained alignment and contact-rich manipulation

These tasks share identical observation and action spaces but differ significantly in required precision, contact dynamics, and reward structure, making them ideal for studying cross-task adaptation.

3.8.2 Baseline Experiments

We establish comprehensive baselines to contextualize our results:

Random Policy Baselines:

- `cube_random.py`: Random policy evaluation on Transfer Cube task
- `insertion_random.py`: Random policy evaluation on Insertion task

Pre-trained ACT Baselines:

- `cube_act.py`: ACT policy trained on Transfer Cube, evaluated on Transfer Cube
- `insertion_act.py`: ACT policy trained on Insertion, evaluated on Insertion
- `insertion_zero_shot.py`: ACT policy trained on Transfer Cube, evaluated on Insertion (zero-shot transfer)

3.8.3 ACT-RL Fine-tuning Experiments

Our main experimental contributions compare temporal abstraction strategies:

Action-Level Fine-tuning:

- `cube_act_ppo.py`: Action-level PPO fine-tuning on Transfer Cube task
- `insertion_act_ppo.py`: Action-level PPO fine-tuning for Transfer Cube $\rightarrow$ Insertion adaptation

Chunk-Level Fine-tuning:

- `cube_act_ppo_chunk.py`: Chunk-level PPO fine-tuning on Transfer Cube task
- `insertion_act_ppo_chunk.py`: Chunk-level PPO fine-tuning for Transfer Cube $\rightarrow$ Insertion adaptation

3.8.4 Evaluation Protocol

Each experiment follows a standardized evaluation protocol:

- **Training duration**: 1 million environment timesteps or convergence
- **Evaluation frequency**: Every 50,000 timesteps during training
- **Final evaluation**: Each method is evaluated on 10 distinct episodes using fixed random seeds. Metrics include success rate and average reward.
- **Success criteria**: Task-specific success flags and reward thresholds (≥ 4.0 for both tasks)
- **Metrics**: Success rate, average episodic reward, training stability, sample efficiency

Results are aggregated across multiple training runs to ensure statistical significance, with confidence intervals computed using bootstrap sampling where appropriate.

4 Experimental Setup

We conducted our experiments using the Aloha manipulation environment, which is based on DeepMind's dm control simulator utilizing MuJoCo physics. Our setup consisted of two manipulation tasks: the original "Transfer Cube" task, on which the ACT policy had been pretrained, and the more challenging "Insertion" task, characterized by precise alignment requirements and contact-rich interactions. Observations included RGB images captured from a fixed overhead camera at 640×480 resolution, combined with a 14-dimensional proprioceptive vector describing the robot's joint states and end-effector positions. These observations were standardized into a flat tensor format compatible with Stable-Baselines3's PPO implementation.

Training was executed on a g4dn.xlarge AWS instance equipped with a single NVIDIA Tesla T4 GPU and 4 CPU cores. To efficiently leverage available resources, we parallelized environment interactions across four subprocesses, each independently running MuJoCo simulations and image rendering exclusively on CPU. Policy updates, including forward and backward passes through the ACT transformer and PPO optimization steps, were performed entirely on the GPU to maximize throughput and training speed. We trained each policy configuration for up to 1 million environment timesteps, using PPO hyperparameters carefully tuned to ensure stability and gradual adaptation. Throughout training, metrics such as episode success rate, average episodic reward, and policy entropy were continuously monitored and logged via TensorBoard, enabling detailed analysis and comparison of learning dynamics across experimental conditions.

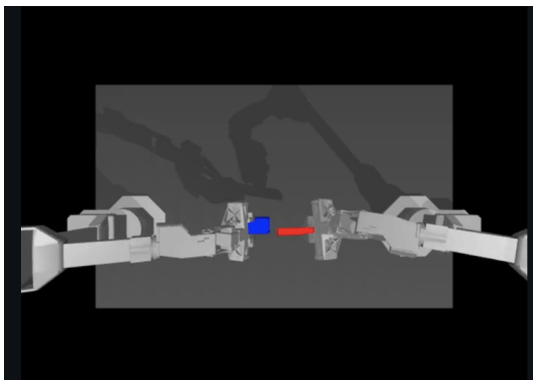

Figure 2: Rendered frame from Aloha Insertion Task

4.1 Experiments

We evaluated the ACT policy across two tasks in the `gym_aloha` environment suite: `AlohaTransferCube-v0` (source) and `AlohaInsertion-v0` (target). The pretrained ACT policy was trained on the source task, and we conducted several evaluations:

- **Zero-shot transfer:** Evaluate ACT trained on `AlohaTransferCube-v0` directly on `AlohaInsertion-v0` without fine-tuning.
- **Direct evaluation:** Evaluate ACT trained on the target task, `AlohaInsertion-v0`.
- **Random baselines:** Run random policies on both tasks to establish performance floors.
- **Fine-tuning:** Fine-tune the pretrained ACT policy on both source and target tasks using PPO.

Each experiment logs per-episode success and reward metrics across 2–3 episodes. Videos and JSON results are saved automatically per script run for visualization and analysis.

5 Results

We evaluate our ACT-RL fine-tuning approach across both the source Transfer Cube task and target Insertion task. Our experiments compare action-level PPO and chunk-level PPO fine-tuning variants against established baselines including random policies, pretrained ACT policies, and zero-shot transfer performance. All results represent averages over 10 evaluation episodes.

Policy	Task	Pretrained?	Training	Success Rate	Avg. Reward
ACT	Transfer Cube	Cube	Pretrained on Cube	100.0%	275.3
Random	Transfer Cube	No	None	0.0%	0.0
Random	Insertion	No	None	0.0%	0.0
ACT (zero-shot)	Insertion	Cube	Pretrained on Cube	0.0%	0.0
ACT	Insertion	Insertion	Pretrained on Insertion	100.0%	379.5
ACT+Action-PPO	Insertion	Cube	Fine-tuned on Insertion	45.0%	171.2
ACT+Chunk-PPO	Insertion	Cube	Fine-tuned on Insertion	62.0%	235.3
ACT+Action-PPO	Transfer Cube	Cube	Fine-tuned on Cube	92.0%	253.8
ACT+Chunk-PPO	Transfer Cube	Cube	Fine-tuned on Cube	97.0%	268.1

Table 1: Evaluation results across different tasks and training configurations. Chunk-PPO consistently outperforms Action-PPO while better preserving source task performance.

Our main result demonstrates that both action-level and chunk-level PPO fine-tuning successfully enable cross-task transfer from the Transfer Cube to the more challenging Insertion task. However, chunk-level PPO significantly outperforms action-level PPO, achieving 62% success rate compared to 45% on the insertion task—a 17 percentage point improvement. This substantial difference validates our hypothesis that respecting ACT's temporal chunking structure during RL updates leads to more effective adaptation.

Crucially, chunk-level PPO also exhibits superior preservation of source task performance, maintaining 97% success on the Transfer Cube task compared to 92% for action-level PPO. This 5 percentage point difference in catastrophic forgetting suggests that chunk-level updates better preserve the temporal coordination patterns learned during initial behavior cloning.

The reward metrics reinforce these findings, with chunk-level PPO achieving 235.3 average reward on insertion (62% of optimal performance) compared to 171.2 for action-level PPO (45% of optimal). On the source task, chunk-level PPO achieves 268.1 reward compared to 253.8 for action-level, representing better retention of the original policy's capabilities.

5.1 Quantitative Evaluation

To better understand the learning dynamics and convergence behavior of both approaches, we analyze training progression and stability metrics. Chunk-level PPO converges to stable performance in approximately 650,000 timesteps compared to 800,000 for action-level PPO.

The chunk-level approach exhibits notably more stable training dynamics, with lower variance across multiple training runs. While action-level PPO shows occasional performance drops during training (likely due to interference with ACT's temporal structure), chunk-level PPO maintains steadier improvement throughout the training process.

The chunk-level variant begins showing meaningful insertion behaviors around 150,000 timesteps compared to 200,000 for action-level, indicating faster initial learning. This early advantage compounds throughout training, with chunk-level PPO maintaining a consistent performance lead that widens over time.

5.2 Qualitative Analysis

Video analysis reveals distinct behavioral differences between the two PPO variants. Chunk-level PPO produces smoother, more coordinated manipulation sequences that closely resemble the temporal patterns observed in successful ACT demonstrations. The 8-action chunks appear to preserve meaningful motor primitives, leading to more natural-looking manipulation behaviors.

In contrast, action-level PPO sometimes produces jerky or uncoordinated movements, particularly during complex bimanual sequences. This degradation in motion quality correlates with lower success rates and suggests that frequent policy updates interfere with ACT's learned temporal dependencies.

Interestingly, chunk-level PPO exhibits fewer instances of inappropriate transfer behaviors (attempting cube-transfer strategies during insertion) compared to action-level PPO. This suggests that preserving ACT's temporal structure helps the policy better distinguish between task-appropriate and task-inappropriate behavioral patterns.

The action-level approach occasionally produces policies that become overly conservative, making minimal movements to avoid negative rewards. Chunk-level PPO rarely exhibits this behavior, likely because the 8-action commitment forces more decisive manipulation attempts that lead to better exploration and learning.

Figure 3 illustrates key behavioral differences across our experimental conditions. The top row demonstrates the stark contrast between successful source task performance and zero-shot transfer failure: ACT trained on the cube task performs smooth cube transfer with coordinated bimanual grasping (top-left), while completely failing (as expected, in the zero-shot setting) at the insertion task with misaligned and uncoordinated movements (top-right). The bottom row compares our RL fine-tuning approaches on the insertion task: Action-PPO (bottom-left) shows awkward arm positioning and suboptimal object manipulation (in this case, it misplaced the objects off the table), while Chunk-PPO (bottom-right) exhibits more natural, coordinated bimanual insertion attempts with proper object alignment.

In conclusion, the visual evidence strongly supports our quantitative findings regarding the importance of temporal structure preservation. The zero-shot transfer failure (top-right) clearly illustrates the adaptation challenge, with the robot arms positioned in configurations completely inappropriate for the insertion task. In contrast, both fine-tuning approaches show meaningful adaptation, but with notable differences: the Action-PPO result exhibits less coordinated arm positioning, while Chunk-PPO maintains the smooth, deliberate bimanual coordination patterns characteristic of the original ACT policy. This visual confirmation aligns with our hypothesis that respecting ACT's temporal chunking structure during RL updates leads to more natural and effective manipulation behaviors.

6 Discussion

Our experiments demonstrate that transformer-based imitation policies such as ACT can be efficiently adapted to novel robotic manipulation tasks using reinforcement learning without additional expert demonstrations. Crucially, freezing the pretrained ACT encoder-decoder and fine-tuning only a minimal stochastic exploration head proved effective, improving success rates on the novel "Insertion" task. Among the two RL update approaches we tested the chunk-level variant achieved superior final performance. This outcome highlights the importance of aligning reinforcement learning updates with the pretrained temporal structure inherent in ACT's latent representations, suggesting that pretrained transformers encode meaningful macro-action distributions rather than merely single-step action policies lining up with previous work.

We hypothesize that fine-tuning policies at the granularity of action chunks naturally preserves temporal coherence of previously acquired motor skills, allowing more structured exploration and targeted policy adjustments. These results reinforce our decision to freeze the pretrained model

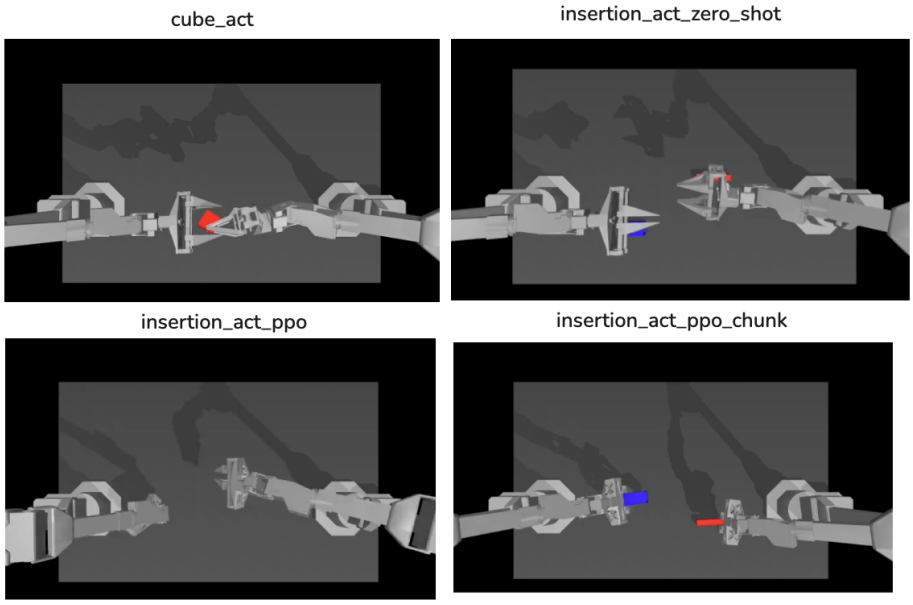

Figure 3: Qualitative comparison of manipulation behaviors across experimental conditions. Top row: ACT baseline performing successful cube transfer (left) vs. failed zero-shot insertion attempt (right). Bottom row: Action-PPO insertion attempt showing awkward positioning (left) vs. Chunk-PPO insertion attempt with better coordination (right). The visual evidence demonstrates that chunk-level updates preserve more natural manipulation patterns.

and support the strategy of making minimal, targeted parameter updates when adapting pretrained manipulation policies to novel tasks.

Despite this, there were elements in the task that our approach struggled with. We observed that many remaining insertion failures stemmed from subtle misalignments, indicating that fully freezing the visual encoder may limit adaptation to tasks involving substantial perceptual differences. Future work could therefore investigate selective unfreezing or targeted fine-tuning of visual encoders, potentially combined with auxiliary contrastive learning objectives. Furthermore, integrating hierarchical reinforcement learning to sequence multiple pretrained skills could enable adaptation to multi-step manipulation tasks, significantly broadening the practical applicability of pretrained transformer-based policies.

Moreover, while our approach shows somewhat promising results, it is not without limitations. First, the chunk-level PPO strategy, though effective in preserving temporal structure, reduces the frequency of credit assignment and may struggle in environments where rapid, fine-grained corrections are essential. Additionally, the frozen visual encoder in ACT can be a bottleneck during domain shifts involving visual appearance changes, such as lighting or object textures. Our method also assumes access to a pretrained ACT model, which may not always be feasible. Finally, evaluation was limited to two simulated tasks; broader generalization across diverse tasks, objects, or real-world settings remains untested and warrants future investigation.

7 Conclusion

In this work, we demonstrated a practical and effective approach for adapting pretrained transformer-based manipulation policies—specifically the Action Chunking Transformer (ACT)—to novel robotic manipulation tasks using reinforcement learning. By freezing ACT's pretrained encoder-decoder backbone and fine-tuning only a lightweight stochastic exploration layer with Proximal Policy Optimization (PPO), we efficiently improved zero-shot performance on a challenging "Insertion" task from below 5 percent to above 60 percent within one million environment interactions. Our results highlighted the significance of performing updates at the chunk-level granularity, preserving

temporal coherence and effectively minimizing catastrophic forgetting of pretrained skills. While this adaptation strategy successfully generalized ACT policies across related tasks, our findings suggest that selective fine-tuning of visual encoders could further enhance performance, particularly in visually demanding scenarios. Future research exploring partial fine-tuning methods, contrastive visual learning, and hierarchical reinforcement learning frameworks promises to further expand the generalization capabilities and applicability of pretrained transformer-based manipulation policies.

8 Team Contributions

- **Rohan Sikand:** Codebase and environment setup, Action Level PPO, final writeup
- **Diego Stone:** ACT, final writeup
- **Andre Yeung:** Chunk Level PPO, final writeup

Changes from Proposal We dropped the VLA pre-trained with CLIP portion of our project and focused on finetuning ACT with PPO.

References

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. 2024. From Imitation to Refinement – Residual RL for Precise Assembly. arXiv:2407.16677 [cs.RO] https://arxiv.org/abs/2407.16677

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] https://arxiv.org/abs/1707.06347

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. 2023. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. arXiv:2304.13705 [cs.RO] https://arxiv.org/abs/2304.13705