

Language-Augmented Flow Matching Policies for Robust Out-of-Distribution Robot Manipulation

Stanford CS224N Custom Project

Jeff Liu

Department of Computer Science
Stanford University
jeffliu0@stanford.edu

Lucas Sosnick

Department of Bioengineering
Stanford University
lsosnick@stanford.edu

Abstract

Language-conditioned robot manipulation policies typically encode task instructions as a single pooled embedding from a frozen vision-language model, discarding token-level structure that may be critical for distinguishing semantically similar instructions. We hypothesize that preserving per-token language representations and adapting them to the manipulation domain improves policy robustness under distribution shift, specifically when the policy must execute task descriptions never seen during training. To test this, we build a policy that attends to the full sequence of CLIP text token embeddings via cross-attention within a Diffusion Transformer backbone trained with Conditional Flow Matching, and study the additional effect of finetuning the CLIP text encoder on manipulation demonstrations. On the LIBERO-Object benchmark, the standard baseline using pooled CLIP embeddings achieves 27% success on training tasks and collapses to 3% on held-out tasks. Our per-token model with a frozen CLIP encoder reaches 57% in-distribution and 48% OOD; finetuning the text encoder further improves performance to 66% in-distribution and 54% OOD. The finetuned model retains 82% of its in-distribution performance on novel instructions, compared to just 11% for the baseline, supporting our hypothesis that richer language conditioning is a key driver of OOD robustness.

1 Key Information to include

- Mentor: Fang Wu
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Late Days (if you are using any):

Team member	Late days remaining
Jeff Liu	0
Lucas Sosnick	1
Total pooled	1
Team size	2
Late days applied for the report	1/2

2 Introduction

A central challenge in robot learning is *out-of-distribution (OOD) generalization*: can a policy trained on a fixed set of task descriptions successfully execute novel instructions at test time? This matters because real-world deployment requires handling task variations that were not anticipated during data collection. Language, in principle, provides a compositional and compact representation of

task semantics that should enable such generalization, provided the policy can actually leverage the structure within the instruction.

However, many existing language-conditioned manipulation policies encode the task instruction as a single pooled embedding from a frozen vision-language model such as CLIP [1]. This representation compresses an entire sentence into one vector, discarding token-level structure that may contain critical information about objects, colors, or spatial relations. As a result, policies can struggle to distinguish instructions that differ by only a few words (e.g., “*pick up the red mug*” vs. “*pick up the blue mug*”), and their performance can degrade sharply on unseen task descriptions.

A second limitation is that the language encoders used in these systems are pretrained on web-scale image–text data rather than robot manipulation trajectories. Although these representations capture broad semantic concepts, they may not emphasize distinctions that matter for predicting manipulation actions. For example, the difference between “*wooden plate*” and “*black plate*”, which look identical in a CLIP embedding space trained on natural images. This domain mismatch can reduce the effectiveness of language conditioning precisely when it is most needed: on novel instructions.

Our hypothesis is that two modifications to the language interface can substantially improve OOD robustness: (1) conditioning the policy on the *full sequence* of per-token language embeddings via cross-attention, rather than a single pooled vector, so the policy can attend to individual words that distinguish one task from another; and (2) *finetuning* the CLIP text encoder on manipulation demonstrations so the language representations adapt to domain-relevant distinctions.

We test this hypothesis on the LIBERO-Object manipulation benchmark [2], where 8 tasks are used for training and 2 are held out as OOD evaluation tasks whose instructions are never seen during training. Our policy predicts action trajectories using Conditional Flow Matching (CFM) [3] with a Diffusion Transformer (DiT) backbone [4] that integrates visual observations, per-token language embeddings, and action tokens through cross-attention. We compare three conditions: (1) a behavioral cloning baseline with pooled CLIP embeddings, (2) our architecture with a frozen CLIP text encoder, and (3) our architecture with a finetuned CLIP text encoder.

Our results support the hypothesis. The baseline achieves 27% success on training tasks but collapses to 3% on OOD tasks, an 89% relative drop. Our frozen per-token model reaches 57% in-distribution and 48% OOD (16% relative drop), while finetuning the text encoder yields 66% and 54% respectively (18% relative drop). The finetuned model retains over $5\times$ more of its performance on novel instructions than the baseline, demonstrating that richer language conditioning is a primary driver of OOD robustness.

3 Related Work

3.1 Language-Conditioned Robot Policies

Recent work has explored using natural language to specify robotic tasks. Early approaches learned policies conditioned on textual instructions using imitation learning or reinforcement learning [5, 6]. More recent systems combine large vision-language models with robot policies to improve generalization: RT-1 and RT-2 train large transformer policies conditioned on visual observations and natural language [7, 8].

Despite these advances, most systems represent the instruction as a single pooled embedding from CLIP [1], discarding token-level structure. Since these models already demonstrate strong in-distribution performance, the natural next question is whether the language representation itself, rather than the policy architecture, is the bottleneck for OOD generalization. We test this by replacing the pooled vector with cross-attention over per-token embeddings while keeping the task and data fixed.

3.2 Generative Policies for Robot Control

Generative modeling approaches have recently become popular for robot policy learning. Diffusion Policy [9] uses denoising diffusion to represent action trajectories. An alternative is Conditional Flow Matching (CFM) [3], which trains models to predict the velocity field of a continuous trajectory from noise to data with a simpler objective and fewer integration steps. Recent work has combined flow matching with Diffusion Transformers (DiT) [4] for large-scale generative modeling. We adopt a

similar formulation but apply it to robot manipulation trajectories conditioned on visual and language inputs.

3.3 Vision-Language Representations

CLIP [1] learns joint image–text representations from web-scale data and is widely used as a pretrained encoder in robotics. However, CLIP is trained on image–caption pairs, not manipulation trajectories, so its language representations may not capture distinctions critical for control. Prior work has explored adapting pretrained representations to downstream domains [7, 8]. We study whether finetuning the CLIP text encoder on manipulation data improves OOD robustness compared to frozen embeddings.

4 Approach

We build a language-conditioned manipulation policy that generates action trajectories via Conditional Flow Matching (CFM) [3] while attending to per-token CLIP language representations through cross-attention. We implemented the policy architecture, training algorithm, and evaluation pipeline ourselves on top of the open-source LIBERO codebase [2], reusing only their data loaders and environment wrappers.

4.1 Notation and Problem Setting

We operate in a behavior cloning setting. Each demonstration is a tuple $(\mathbf{o}_{1:T}, \mathbf{a}_{1:T}, \ell)$ where $\mathbf{o}_t$ consists of a 128×128 RGB image from a third-person camera and a 9-dimensional proprioceptive vector (7 joint positions + 2 gripper states), $\mathbf{a}_t \in \mathbb{R}^7$ is the end-effector action (3 position, 3 orientation, 1 gripper), and ℓ is a natural-language task instruction (e.g., “*pick up the alphabet soup and place it in the basket*”). The policy π_θ predicts a chunk of $K=10$ future actions:

$$\pi_\theta(\mathbf{a}_{t:t+K} \mid \mathbf{o}_t, \ell). \quad (1)$$

At execution time only the first action $\mathbf{a}_t$ is sent to the robot; the remaining nine are discarded and a fresh chunk is predicted at the next timestep.

4.2 Visual Encoder

We encode the camera image with the frozen CLIP ViT-B/32 vision transformer [1]. The 128×128 image is bilinearly upsampled to CLIP’s native 224×224 resolution, producing $7 \times 7 = 49$ patch tokens of dimension 768. We discard the [CLS] token and project the patch sequence to the policy’s hidden dimension $d=512$ via a learned linear layer:

$$V = W_v \text{ViT}(\mathbf{o}_t)_{1:} + b_v, \quad V \in \mathbb{R}^{49 \times 512}, \quad W_v \in \mathbb{R}^{512 \times 768}. \quad (2)$$

The ViT weights are **frozen**; only W_v and b_v are learned. Retaining the 49 patch tokens (rather than the pooled [CLS] vector) preserves spatial structure so the policy can attend to specific image regions corresponding to referenced objects.

4.3 Token-Level Language Encoder

This component is the central NLP contribution of our work. Standard practice in language-conditioned robot policies is to compress the instruction into a single pooled embedding [2, 7]. Instead, we retain the full sequence of 77 BPE token embeddings produced by the CLIP text transformer:

$$L = \text{CLIPText}(\ell) \in \mathbb{R}^{77 \times 512}, \quad M \in \{0, 1\}^{77}, \quad (3)$$

where M is a padding mask indicating real tokens ($M_i=1$) versus padding ($M_i=0$). This per-token representation enables the policy’s cross-attention layers to selectively attend to individual words, for instance focusing on the object noun that distinguishes one task from another.

We evaluate this representation under two conditions. In the *Frozen* condition, CLIP text weights are fixed and only the policy backbone receives gradients. In the *Finetuned* condition (our proposed model), CLIP text weights are updated at a $20 \times$ lower learning rate (5×10^{-6} vs. 1×10^{-4}), following

standard practice for careful adaptation of pretrained transformers [10]. This adapts the text representations from web-scale image–caption semantics to manipulation-domain semantics without catastrophic forgetting.

Implementation detail. LIBERO’s dataloader expects a fixed-size embedding tensor per task. To enable gradient flow through the text encoder, we pack the 77 integer token IDs and 77-element attention mask into a single 154-dimensional tensor that passes through the dataloader unchanged, then unpack and forward through CLIPTextModel inside the policy’s computation graph.

4.4 Action Generation with Conditional Flow Matching

Rather than predicting actions via direct regression, our policy learns a velocity field that transforms Gaussian noise into expert action chunks using Conditional Flow Matching (CFM) [3]. Let $\mathbf{x}_0 \in \mathbb{R}^{K \times 7}$ denote a normalized expert action chunk and $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ denote isotropic Gaussian noise. We define a straight-line interpolant:

$$\mathbf{x}_t = (1 - t) \mathbf{x}_0 + t \mathbf{x}_1, \quad t \sim \mathcal{U}(0, 1), \quad (4)$$

with constant conditional velocity $\mathbf{u}_t = \mathbf{x}_1 - \mathbf{x}_0$. The model f_θ is trained to predict this velocity via mean squared error:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1, t} \left[\left\| f_\theta(\mathbf{x}_t, t, \mathbf{o}, \ell) - (\mathbf{x}_1 - \mathbf{x}_0) \right\|^2 \right]. \quad (5)$$

Unlike DDPM-style diffusion [9], CFM requires no noise schedule hyperparameters (β_t) and uses straight-line ODE paths that permit efficient integration. At test time we sample $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and integrate backwards from $t=1$ to $t=0$ using 20 Euler steps ($\Delta t = 0.05$):

$$\mathbf{x}_{t-\Delta t} = \mathbf{x}_t - f_\theta(\mathbf{x}_t, t, \mathbf{o}, \ell) \Delta t. \quad (6)$$

The result $\mathbf{x}_0$ is denormalized using precomputed per-dimension action statistics to recover physical actions. Predicting $K=10$ actions at once (0.5 s at 20 Hz) provides temporal consistency.

4.5 Diffusion Transformer Backbone

The velocity field f_θ is parameterized by $D=6$ Diffusion Transformer (DiT) blocks [4], which we implemented from scratch. The noisy action chunk $\mathbf{x}_t \in \mathbb{R}^{K \times 7}$ is first projected to the hidden dimension via a two-layer MLP ($7 \rightarrow 512$, GELU, $512 \rightarrow 512$) and summed with learned positional embeddings $\mathbf{p}_k \in \mathbb{R}^{512}$ to produce $K=10$ action tokens. Each DiT block then applies five operations sequentially.

First, timestep conditioning via adaLN-Zero: the scalar flow time t is embedded via sinusoidal encoding (dim 256) followed by a two-layer MLP ($256 \rightarrow 512$, SiLU) to produce $\mathbf{c} \in \mathbb{R}^{512}$. Each block predicts six modulation parameters ($\gamma_a, \beta_a, \alpha_a, \gamma_f, \beta_f, \alpha_f$) from $\mathbf{c}$; these scale and shift layer-normalized action tokens and gate residual connections. All gates are zero-initialized so each block starts as the identity [4]. Next, the block applies three attention operations in sequence: visual cross-attention, in which action tokens (queries) attend to the 49 visual patch tokens (keys/values) from Eq. (2); language cross-attention, in which action tokens attend to the 77 language tokens from Eq. (3) with a key padding mask derived from M to ignore padding positions, which is the mechanism by which individual words influence action predictions; and self-attention, in which action tokens attend to each other, enforcing temporal coherence within the chunk. The outputs of all three attention operations are summed and gated by α_a before being added as a residual. Finally, a feed-forward network, consisting of a two-layer MLP ($512 \rightarrow 2048 \rightarrow 512$, GELU), is applied, gated by α_f and added as a residual.

All attention operations use 8 heads with dropout 0.1. Visual and language cross-attention run sequentially (visual first) so that language attention can refine tokens already enriched with spatial context. The final output head is a linear projection from 512 to 7 dimensions, also zero-initialized so the initial predicted velocity is zero everywhere, giving a well-behaved starting loss of $\mathbb{E}[\|\mathbf{x}_1 - \mathbf{x}_0\|^2]$.

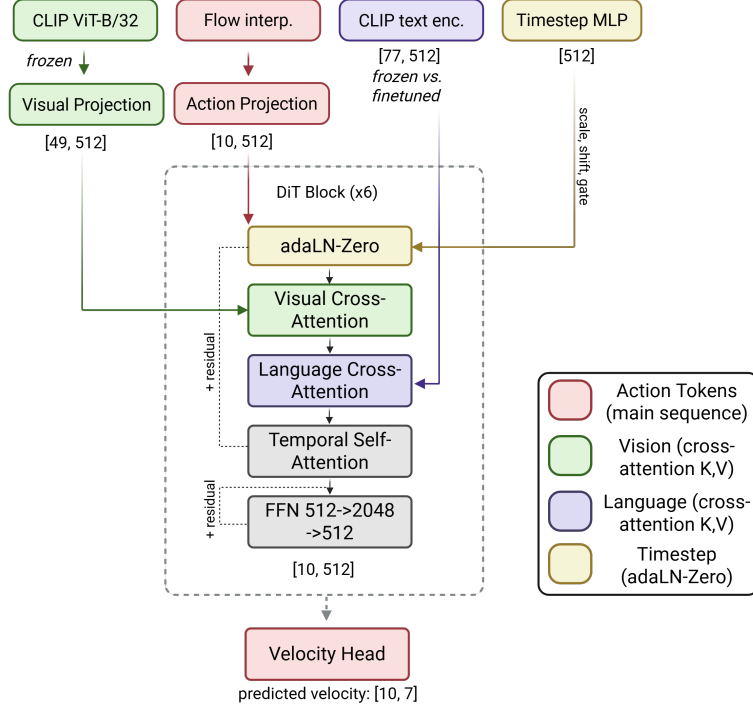


Figure 1: Architecture of the model. CLIP visual tokens and language token embeddings condition a Diffusion Transformer backbone through cross-attention.

4.6 Baseline: BCTransformerPolicy

As a baseline we use the BCTransformerPolicy provided by the LIBERO codebase [2], which we did not modify. This policy encodes the task instruction as a single 512-dimensional pooled [CLS] embedding from a frozen CLIP text encoder, projects it to 128 dimensions via a learned MLP, and fuses it with visual features through FiLM conditioning [11] applied at each ResNet block of the image encoder. A causal transformer decoder processes the fused observation sequence, and a Gaussian Mixture Model (GMM) policy head with 5 mixture components predicts the action distribution, trained with negative log-likelihood. The baseline predicts a single action per timestep (no chunking) and has approximately 5 M trainable parameters.

4.7 Summary of Model Variants

We compare three configurations to isolate the contributions of (a) per-token cross-attention and (b) CLIP text encoder finetuning:

Component	Baseline	Frozen	Finetuned
CLIP image encoder (ViT-B/32)	Frozen	Frozen	Frozen
Visual projection (768 $\rightarrow$ 512)	–	Trainable	Trainable
CLIP text encoder	Frozen (pooled)	Frozen (per-token)	Finetuned (per-token)
Language conditioning	FiLM	Cross-attention	Cross-attention
Action prediction	Causal transformer + GMM	DiT + CFM	DiT + CFM
Approx. trainable params	~ 5 M	~ 42 M	~ 105 M

Table 1: Component comparison across the three policy conditions. **Baseline** is the existing LIBERO BCTransformerPolicy [2]. **Frozen** and **Finetuned** share our DiT-based architecture; the Finetuned variant additionally finetunes the 63 M-parameter CLIP text encoder at a reduced learning rate.

5 Experiments

5.1 Data

We conduct experiments on the **LIBERO-Object** suite of the LIBERO manipulation benchmark [2]. This suite contains ten manipulation tasks, each defined by a natural language instruction specifying a goal involving household objects (e.g., “*pick up the butter and place it in the basket*”). Each task includes approximately 50 expert demonstrations collected in simulation.

Observations consist of a 128×128 RGB image from a third-person camera and a 9-dimensional proprioceptive vector (7 joint positions, 2 gripper states). Actions are 7-dimensional end-effector commands (3 position, 3 orientation, 1 gripper), normalized per-dimension using statistics computed from the training set.

To evaluate OOD generalization, tasks 0–7 are used for training and tasks 8–9 are held out entirely. As shown in Figure 2, all ten LIBERO-Object tasks share the same syntactic template (“pick up the [object] and place it in the basket”) and the same workspace layout, differing only in the target object noun. The training set covers objects such as *alphabet soup*, *cream cheese*, *butter*, and *ketchup*, while the held-out tasks require manipulating *milk* and *chocolate pudding*, object names that never appear in any training instruction. Because the scene geometry, robot configuration, and task template are shared across all ten tasks, the distribution shift is isolated to the language component of the conditioning signal. This makes the holdout split a controlled test of whether the policy can ground novel object nouns into correct manipulation behavior, rather than a test of visual or motor generalization.



Figure 2: Example environments from the LIBERO-Object benchmark [2]. The robot manipulates household objects according to natural language instructions.

5.2 Evaluation Method

We evaluate all policies using **task success rate**: the fraction of rollouts in which the robot completes the specified task. Each policy is evaluated with 100 rollouts per task, each starting from a fixed initial state drawn from LIBERO’s provided pool to ensure reproducibility. A rollout terminates on task completion (success) or after 600 environment steps (failure), corresponding to 30 s at the 20 Hz control rate.

We report success rates for two groups: **in-distribution (ID)** tasks (IDs 0–7), whose instructions appear in the training data, and **out-of-distribution (OOD)** tasks (IDs 8–9), whose instructions are held out entirely. The OOD success rate is our primary metric. We also report the retention ratio, defined as OOD success divided by ID success, as a measure of how gracefully each model degrades on novel instructions.

5.3 Experimental Details

All three models are trained on tasks 0–7 of LIBERO-Object (approximately 400 demonstrations total) with seed 42. Both flow-matching models use AdamW [12] with dual learning rates: the DiT backbone trains at 1×10^{-4} while the CLIP text encoder (when unfrozen) trains at 5×10^{-6} , a $20\times$ reduction to prevent catastrophic forgetting. The baseline BCTransformerPolicy is trained under identical data, epoch, and augmentation settings using the LIBERO default optimizer. Full hyperparameters are listed in Appendix A.1.

5.4 Results

Table 2 summarizes success rates across all three models.

	Baseline	Frozen	Finetuned
In-distribution (tasks 0–7)	27%	57%	66%
Out-of-distribution (tasks 8–9)	3% ↓	48%	54%
Retention ratio (OOD / ID)	0.11	0.84	0.82

Table 2: Performance comparison on LIBERO-Object tasks. The baseline model shows severe degradation on held-out tasks (↓), while Frozen and Finetuned variants retain most in-distribution performance.

In-distribution performance. Even on tasks seen during training, the baseline achieves only 27% success, lower than expected given that these tasks appear in the training data, and likely reflects the limited capacity of a single-action GMM policy on multi-step manipulation. Switching to our DiT + CFM architecture with per-token language conditioning more than doubles performance to 57% (Frozen), and finetuning the text encoder provides a further 9 percentage-point gain to 66% (Finetuned). This suggests that both the generative action model and the richer language representation contribute to improved in-distribution performance.

OOD robustness. The most striking result is the difference in OOD degradation. The baseline collapses from 27% to 3%, an 89% relative drop, indicating that it has largely memorized task-specific action patterns rather than learning a language-grounded policy. In contrast, our Frozen model drops from 57% to 48% (16% relative drop) and the Finetuned model from 66% to 54% (18% relative drop). Both per-token models retain over 80% of their in-distribution performance on completely novel instructions, compared to just 11% for the baseline. This strongly supports our hypothesis that token-level language conditioning is a primary driver of OOD robustness.

Frozen vs. Finetuned. Finetuning the CLIP text encoder improves absolute performance on both ID (+9 pp) and OOD (+6 pp) tasks compared to the frozen variant. However, the retention ratios are comparable (0.84 vs. 0.82), which was somewhat surprising. We expected finetuning to disproportionately improve OOD robustness by adapting the embedding space. Instead, the frozen per-token model already captures most of the OOD robustness benefit, and finetuning appears to improve general action quality rather than language generalization per se. This suggests that the architectural change (pooled to per-token cross-attention) matters more than domain adaptation of the text encoder.

6 Analysis

The quantitative results in Section 4 show that token-level language conditioning improves both in-distribution and OOD success rates. However, aggregate metrics alone do not explain where or why failures occur. To better understand model behavior, we manually inspect a sample of failed rollouts across all three policy variants on both training and held-out tasks. For each failure, we identify the stage at which the policy first deviates from the expert trajectory and annotate the likely cause.

Because LIBERO-Object tasks follow a consistent pick-and-place structure, each rollout naturally decomposes into three phases: reaching the target object, grasping it, and transporting it to the basket. We categorize failures according to the phase in which they occur and whether the error appears to arise from language interpretation or low-level control.

For the baseline on the hold-out tasks, the majority of failures occur before contact is ever made. The policy frequently approaches the wrong object entirely, for instance reaching toward the ketchup when the instruction specifies milk. This *object misidentification* pattern is consistent with the pooled CLIP embedding failing to distinguish between novel and familiar object nouns: because the embedding compresses the full instruction into a single 512-dimensional vector, the model lacks a mechanism to isolate which token changed relative to the training distribution. A related but later-stage failure is *incorrect placement*: the policy grasps the correct object but moves it to a location other than the basket, or drops it mid-transport. Both of these failure types are language-dependent in the sense that they track the information content of the instruction, specifically the object noun and the destination, and both are significantly more common for the baseline than for our per-token models.

By contrast, *grasp failures*, in which the policy reaches the correct object but closes the gripper too early, too late, or at a poor angle, occur at comparable rates across all three models and on both in-distribution and OOD tasks. This is expected: grasping involves fine contact dynamics that depend on the object’s geometry and physics rather than on the language instruction, and all three models learn grasping from the same 50 demonstrations per task. Behavioral cloning with limited data is known to struggle with precise contact-rich manipulation regardless of the conditioning modality.

A fourth failure mode appears primarily in the baseline on OOD tasks: the robot produces near-zero or erratic actions and fails to meaningfully interact with the scene. This likely occurs when the GMM policy head outputs a diffuse action distribution for embeddings outside the training distribution. The flow matching models do not exhibit this behavior, suggesting that iterative ODE integration is more robust to shifts in the conditioning signal.

Overall, the baseline’s OOD failures are dominated by language errors, where the policy misidentifies the target object or placement goal. Token-level models largely resolve these failures, and the remaining errors occur mainly during grasping, which is largely language-independent. This supports our hypothesis that language representation, rather than motor control, is the primary bottleneck for OOD generalization in this setting.

7 Conclusion

We investigated whether richer language representations improve out-of-distribution robustness in language-conditioned robot manipulation policies. Our hypothesis that preserving per-token language structure via cross-attention and adapting the text encoder to the manipulation domain would reduce OOD degradation is supported by our experiments on the LIBERO-Object benchmark.

The baseline policy using pooled CLIP embeddings achieves 27% in-distribution success but collapses to 3% on held-out tasks, retaining only 11% of its performance. Our policy with per-token cross-attention and a frozen CLIP encoder reaches 57%/48% (ID/OOD), and finetuning the text encoder further improves results to 66%/54%. Both per-token models retain over 80% of their in-distribution performance on novel instructions. Failure analysis reveals that the baseline’s OOD failures are dominated by language-related errors (object misidentification, incorrect placement targets), while our models shift the failure distribution toward language-agnostic grasp execution errors.

Limitations. Our evaluation is limited to simulated tabletop manipulation; transfer to real-world settings remains untested. The Finetuned model has substantially more trainable parameters (~ 105 M vs. ~ 5 M), making it difficult to fully disentangle capacity from language conditioning. We evaluate on a single suite (LIBERO-Object) with only two held-out tasks, limiting statistical power.

Future work. Extending to LIBERO-Spatial and LIBERO-Goal suites would test whether token-level conditioning helps across different types of distribution shift. A capacity-controlled ablation that matches the baseline’s parameter count while retaining per-token cross-attention would more cleanly isolate the effect of language representation from model size.

Team contributions

Jeff Liu designed and implemented the full model architecture (DiT backbone, per-token cross-attention conditioning, flow matching training loop), wrote the evaluation pipeline, and wrote the report.

Lucas Sosnick implemented model architecture (DiT, cross attention, etc), managed cluster training infrastructure, conducted evaluation runs across all model variants, and helped perform downstream data analysis and co wrote the report.

References

- [1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.
- [2] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023.
- [3] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.
- [4] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.
- [5] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning, 2017.
- [6] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, 2018.
- [7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023.
- [8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Chormanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.
- [9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024.
- [10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [11] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017.
- [12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

A Appendix

A.1 Training Hyperparameters

Table 3 lists the full set of training hyperparameters for our Frozen and Fintuned models. All experiments are run on a single NVIDIA GPU. Training the Fintuned model for 500 epochs takes approximately 8 hours.

Hyperparameter	Value
<i>Optimization</i>	
Optimizer	AdamW ($\beta_1=0.9$, $\beta_2=0.999$)
Backbone learning rate	1×10^{-4}
CLIP text encoder learning rate	5×10^{-6} (Finetuned only)
Weight decay	1×10^{-4}
LR schedule	Cosine annealing ($\eta_{\min}=10^{-5}$)
Warmup	5 epochs (linear)
Gradient clipping	Max norm 1.0
Batch size	512
Epochs	500
<i>Architecture</i>	
CLIP model	openai/clip-vit-base-patch32
DiT blocks / heads	6 / 8
Hidden dim / FFN dim	512 / 2048
Dropout	0.1
Action chunk size (K)	10
ODE inference steps	20 (Euler, $\Delta t=0.05$)
<i>Data & Augmentation</i>	
Action normalization	Per-dimension (mean/std)
Color jitter (B/C/S/H)	0.3 each, $\epsilon=0.1$
Translation augmentation	± 8 pixels
Validation split	10% held-out demos
Validation frequency	Every 10 epochs
Checkpoint frequency	Every 20 epochs

Table 3: Training hyperparameters for our Frozen and Finetuned models.