arXiv:2607.18236v1 [cs.RO] 20 Jul 2026

Patch Policy: Efficient Embodied Control via Dense Visual Representations

Gaoyue Zhou[1*] **Zichen Jeff Cui**[1*] **Ada Langford**[1] **Bowen Tan**[1]
Yann LeCun[1,3] **Lerrel Pinto**[1,2]

[1]Courant Institute, New York University [2]Meta-FAIR [3]AMI Labs

Abstract: Pretrained dense visual features from Vision Transformers (ViTs) are powerful yet have been underutilized in robot learning. Modern robot policies either compress each observation into a single global token, or rely on visual backbones trained from scratch, sacrificing both fine-grained spatial detail and the benefits of large-scale visual pre-training. While there exist policies that do operate on dense patch features like large vision-language-action models (VLAs), they tend to be heavy and slow, inheriting the full cost of a billion-parameter vision-language model (VLM) backbone. We close this gap with PATCH POLICY, a minimal architectural extension that enables transformer-based policies to consume dense pre-trained patch tokens directly without the computational overhead of a full VLM. At its core is a block-causal attention mask that preserves the temporal causality of standard policies while letting the model attend over many patch tokens per observation, alongside other state information. PATCH POLICY is lightweight, fast, and highly effective. Across four simulated and three real-world environment suites, our method achieves a 40% relative improvement over policies using state-of-the-art global-pooled representations. Furthermore, it surpasses fine-tuned OpenVLA-OFT by 18% while using roughly 0.7% of the parameters. We believe PATCH POLICY provides a pipeline for the robotics community to readily leverage continuing progress in visual representation learning, without sacrificing the training efficiency or inference speed required for high-frequency, reactive control. Videos can be viewed at `https://patch-policy.github.io`.

Keywords: Imitation Learning, Visual Representation

1 Introduction

Vision Transformers (ViTs) [1] have become the de facto standard backbone for computer vision. By processing images as sequences of localized patches, ViTs extract rich, dense representations that preserve fine-grained spatial and semantic details. This architectural shift, especially when combined with large-scale self-supervised and language-image pre-training [2, 3, 4, 5], has driven state-of-the-art results across a vast array of tasks [4, 1, 6, 7, 8, 9, 10, 11, 12, 13, 14]. Many of these visual capabilities, particularly fine-grained geometric understanding and robust feature localization, are directly useful for precise robotic manipulation.

Despite this clear advantage, the integration of dense ViT representations into robot learning has largely lagged behind, leaving the field bifurcated into two suboptimal paradigms. One approach, rooted in classical control and reinforcement learning, is to define the environment observation as a flat state vector. Visual observations are compressed into a global feature vector, derived from ResNet [15] pooled features, or ViT `CLS` tokens. This approach creates a severe informational bottleneck: aggressive spatial compression destroys the fine-grained, local details necessary for precise manipulation. A newer approach is to fine-tune large vision-language models (VLMs) on action

*Equal contribution. Corresponding author: `gz2123@nyu.edu`

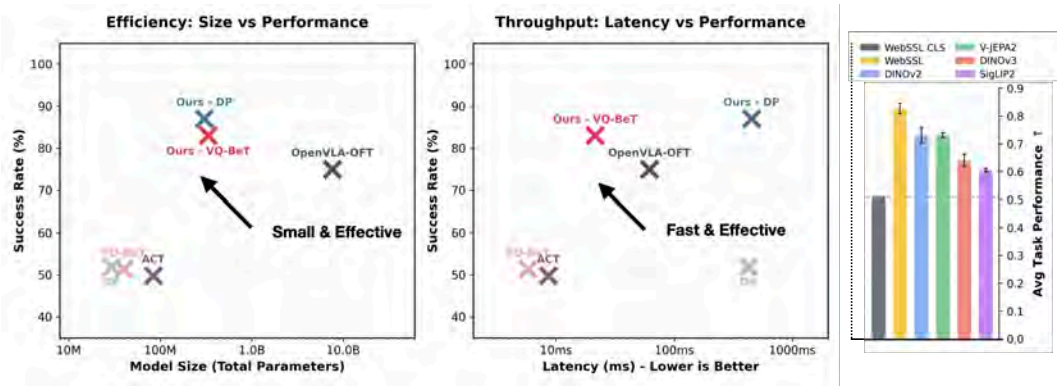

Figure 1: We introduce PATCH POLICY, an efficient policy architecture that harnesses the power of pre-trained dense visual features. PATCH POLICY demonstrates superior performance while remaining computationally lean in both parameter count and inference latency. Through a rigorous analysis across five state-of-the-art visual representations, we show that our dense patch-based approach provides consistent performance gains over traditional global-feature policy baselines (WebSSL `CLS`).

data. While these vision-language-action models (VLAs) have demonstrated strong performance across benchmarks, practitioners often have to fine-tune massive VLM backbones on their specific robot embodiments and downstream tasks for optimal performance, lugging around billions of parameters. Some optimization techniques improve the situation [16, 17], but training and inference remain quite expensive overall.

Can we inherit the representational gains of large-scale vision pretraining, without inheriting the cost of billion-parameter generative models? We argue for an efficient alternative: **visuomotor policies simply need dense features**. The dense visual understanding that makes VLAs effective is already present in Internet-scale pretrained ViTs, available off-the-shelf. By replacing global-pooled features with these patch features, PATCH POLICY captures the spatial detail relevant for precise manipulation at a fraction of the cost. We show that:

1. **Dense representations outperform global features for control**: on precise, multi-object/spatial tasks, spatially dense ViT patches substantially outperform global pooled features or `CLS` tokens while remaining competitive on the rest, regardless of the chosen policy architecture (Table 1, Table 5, Table 6).
2. **Pretrained ViT features transfer to control off-the-shelf**: frozen patch features from Internet-scale ViTs [18], with no encoder fine-tuning, yield robust representations for control (Table 1, Fig. 5, Table 7, Table 8), and enable real-world precise manipulation (Table 2, Table 5).
3. **Spatial compression degrades control performance**: reducing spatial resolution, whether via pooling, or learned convolutional compression, degrades performance (Table 1, Table 4).
4. **PATCH POLICY is highly efficient**: PATCH POLICY matches or outperforms a large VLA fine-tuned on downstream tasks using 0.7% of its parameter count, and runs at as low as $\sim$ 11ms inference latency (Table 3, Section 3.6).

All code and data will be open-sourced. All hyperparameters and implementation details are in Section A.1 and Section A.5 for reproducibility.

2 PATCH POLICY

PATCH POLICY is a modular, Transformer-based [19] policy architecture designed to directly ingest uncompressed, dense patch features. It avoids the traditional global pooled feature bottleneck and preserves the fine-grained spatial information critical for precise manipulation. Our approach is compatible with a wide range of visual encoders and transformer-based policy architectures. The framework consists of an observation trunk (Section 2.1), which encodes sensory inputs and goal

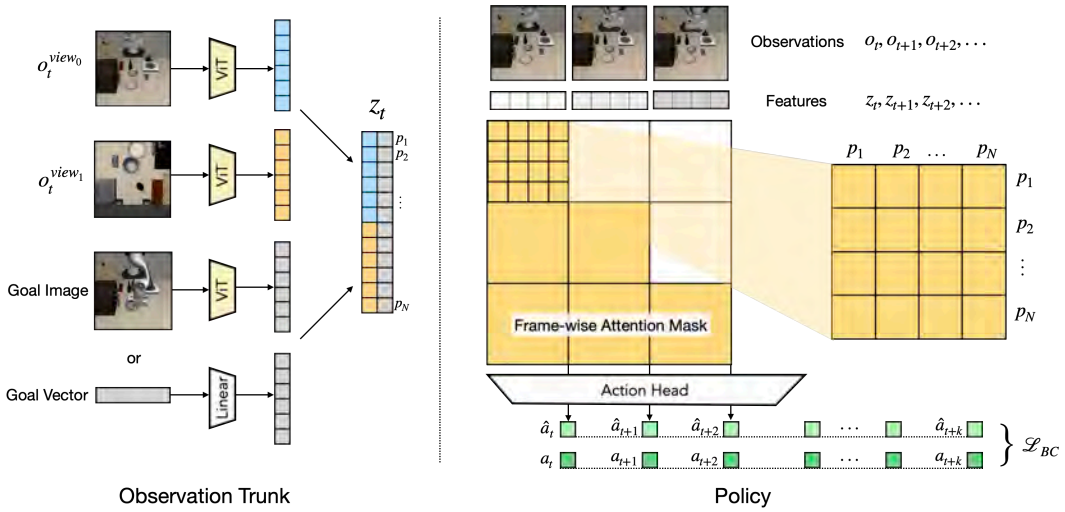

Figure 2: **Architecture**. PATCH POLICY consists of an observation trunk (left) with a policy head (right). We encode multiview observations into patch features and optionally concatenate goal embeddings (images/states) into the current timestep sequence. PATCH POLICY is compatible with any transformer-based policy head. A block-wise causal mask is applied to enforce conditioning on past observations, allowing the policy to be optimized with any standard action head.

specifications into a sequence of dense spatio-temporal features, and a policy head (Section 2.2) that processes this sequence to emit actions.

2.1 Observation Trunk

Given an image observation $o_t \in \mathbb{R}^{C\times H\times W}$, a ViT encoder divides it into patches and extracts dense patch features of shape $P \times D$ (number of patches $\times$ patch embedding dimension). We simply take all patch features as the visual representation for downstream policy learning. For an observation context window of length T, the resulting feature is a sequence of patch features of shape $T \times P \times D$. This formulation is agnostic to the specific ViT architecture or pretraining objective, and is backwards-compatible with global pooled features or state-based environments by setting $P = 1$.

For goal-conditioned behavioral cloning, we admit either a goal image or a goal vector input. For goals specified as an image, we encode it with the same encoder and concatenate it with the observation tensor to form a policy input tensor z_t of shape $T \times P \times 2D$. For goals specified as a vector $g \in \mathbb{R}^G$, we concatenate it to every observation token to form a tensor of shape $T \times P \times (D + G)$. See Fig. 2 for a detailed illustration.

2.2 Policy Learning

We formulate policy learning as a sequence modeling problem over the extracted patch features. PATCH POLICY is compatible with any transformer-based policy architecture taking sequential inputs. To process the spatio-temporal patch feature tensor, we flatten the features into a sequence of length $T \times P$, add a learned 1D positional embedding indexed by the token's position in the flattened sequence, and apply a block-causal attention mask: patches maintain full bidirectional attention intra-frame but are causally masked inter-frame, allowing the model to integrate spatial information across each frame while preserving temporal causality, similar to [20]. We ablate this choice in Section A.6.1. Finally, the policy processes the masked sequence and emits a predicted action chunk at the last patch token for each frame with an action head. See Fig. 2 for details.

This formulation is agnostic to the action head architecture and training objective. In our experiments, we evaluate PATCH POLICY using two state-of-the-art architectures: Vector-Quantized Be-

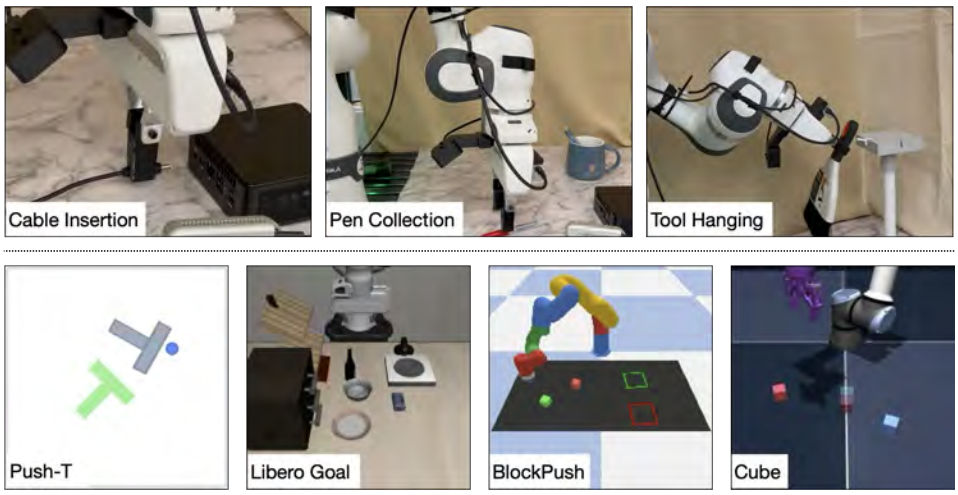

Figure 3: We evaluate PATCH POLICY on four simulated and three real-world environments.

havior Transformer (VQ-BeT) [21], which uses a hybrid classification-regression loss, and Diffusion Policy (DP) [22], which uses a denoising objective.

During training, we forward a sequence of patch tokens through the policy transformer trunk and action head, and compute a loss between the predicted and ground-truth actions for each frame. For inference, we extract the patch features from the current observation and append them to a rolling context window of length T. The policy predicts a chunk of actions from the observation context, and we execute the actions with receding horizon control.

3 Experiments

We evaluate PATCH POLICY across four simulated environments and three real robot manipulation environments, aiming to answer the following key research questions:

1. How does PATCH POLICY compare to state-of-the-art policies using global and patch features?
2. Does PATCH POLICY work for real-world precise manipulation?
3. How do the latest pretrained encoders perform as representations for downstream policy learning?
4. How does the spatial compression of visual features impact downstream task performance?
5. How does PATCH POLICY compare to other methods in terms of efficiency? How do attention design and model size affect its performance? (Ablations Section A.6)

3.1 Environments

We evaluate PATCH POLICY across four simulated environments (Push-T, LIBERO Goal, Block-Push, Cube) with 2D-to-7D action spaces, and three real-world tasks using a 7-DoF Franka arm with a parallel-jaw gripper (inserting a power cable, hanging a tool, and collecting pens into a holder). Environments are visualized in Figure 3 and detailed in App. Section A.2.

3.2 Baselines

We compare against two groups of baselines. To isolate the effect of patch features, we pair the same policy heads (VQ-BeT, Diffusion Policy) with standard global representations: DynaMo [23], `CLS` tokens, and global average pooling. To position PATCH POLICY against the state of the art, we compare to ACT [24] and OpenVLA-OFT [25], which also consume dense features. Our baseline representations and policies are as follows:

1. **Visual Representation Baselines:** To evaluate different feature extraction strategies, we compare against:
   - **DynaMo [23]:** A global pooled representation learned via dynamics-based joint-embedding predictive architecture.
   - `CLS` **Tokens:** The class token from the Vision Transformer, representing a compressed summary of the scene.
   - **Average Pooling:** A baseline that collapses the spatial feature map into a single vector via global average pooling.
2. **ACT [24]:** A conditional VAE that predicts action chunks with temporal ensembling over overlapping chunks, reducing compounding error in fine manipulation. It uses patch features from a ResNet-18 vision encoder trained from scratch.
3. **OpenVLA-OFT [25]:** A Vision-Language-Action (VLA) model that finetunes OpenVLA with parallel action decoding and L1 action regression for faster training and inference. It builds on a Llama-2 7B LLM backbone and consumes channel-fused patch features from pretrained DINOv2 [6] and SigLIP [26] vision encoders.

3.3 How well does PATCH POLICY work?

Table 1: **PATCH POLICY on simulated environments.** *Top*: standard policies that consume globally pooled visual features. *Middle*: our pipeline with WebSSL patch features. *Bottom*: other policies that consume patch tokens. PATCH POLICY substantially improves over global-feature counterparts of the same policies, while matching or exceeding patch-based baselines.

Visual Representation	Policy	Push-T	LIBERO Goal	BlockPush	Cube
Standard policies with globally pooled visual features					
DynaMo	VQ-BeT	0.66	0.93	0.65	0.28
WebSSL `Avg Pool`	VQ-BeT	0.54 ±0.02	0.97 ±0.04	0.84 ±0.18	0.25 ±0.02
WebSSL `CLS`	VQ-BeT	0.59 ±0.01	0.95 ±0.01	0.77 ±0.08	0.23 ±0.01
DynaMo	Diffusion Policy	0.73	0.68	1.06 ±0.10	0.27
WebSSL `Avg Pool`	Diffusion Policy	0.79 ±0.02	0.98 ±0.01	1.34 ±0.02	0.21 ±0.03
WebSSL `CLS`	Diffusion Policy	0.68 ±0.02	**0.99** ±0.01	0.99 ±0.12	0.21 ±0.03
PATCH POLICY: *patch features*					
WebSSL Patch	VQ-BeT **(Ours)**	0.68 ±0.03	0.94 ±0.01	**1.68** ±0.15	1.68 ±0.03
WebSSL Patch	Diffusion Policy **(Ours)**	**0.80** ±0.01	0.98 ±0.00	1.65 ±0.08	**1.73** ±0.02
Other patch-based policies (baselines)					
ResNet-18 Patch	ACT	0.64 ±0.03	0.93 ±0.02	0.15 ±0.01	0.69 ±0.11
DINOv2 + SigLIP Patch	OpenVLA-OFT	0.59 ±0.02	0.95	1.43 ±0.17	1.50 ±0.09

We evaluate PATCH POLICY on four simulated manipulation benchmarks (Push-T, LIBERO Goal, BlockPush, and Cube), spanning planar pushing, goal-conditioned multi-task manipulation, and multi-stage tasks. Table 1 organizes the comparison along the two axes that mirror our contribution: against the same policy classes consuming globally pooled visual features (top), and against other transformer-based policies that already consume patch tokens (bottom).

We report WebSSL [12] results in Table 1 for its strong performance among the encoders we evaluate. A full comparison across encoders is provided in Section 3.5. All policies receive only visual inputs. For the global representation baseline, we evaluate two variants commonly used in policy training: the `CLS` token and `Avg Pool` (global average pooling across all patch features). To maintain consistency across policies, we remove proprioceptive inputs from ACT's CVAE encoder, conditioning its latent variable z on action chunk alone. We also add support for channel-stacking goal conditioning in ACT experiments, mirroring VQ-BeT and Diffusion Policy. We evaluate 100 trajectories per seed for each environment, reporting target coverage (Push-T), success rate (LIBERO Goal), or the average number of objects placed at the goal (BlockPush, Cube). We select the best-performing checkpoint for each run. For the OpenVLA-OFT baseline on LIBERO Goal, we report the results directly as provided in the original manuscript [25].

In Table 1, PATCH POLICY using WebSSL patch features consistently outperforms global representations like DynaMo [23] on VQ-BeT and Diffusion Policy heads on precise, multi-object/spatial tasks (BlockPush, Cube), and competitive elsewhere. Replacing patches with a `CLS` token or average pooling (`Avg Pool`) severely degrades performance; the former favors high-level semantics over spatial details, while the latter lacks the resolution of uncompressed patches. Remarkably, PATCH POLICY outperforms the fine-tuned OpenVLA-OFT baseline which fuses both DINOv2 and SigLIP features on all four environments, confirming that dense spatial representations are vital for complex manipulation tasks.

3.4 Real World Robotic Manipulation with PATCH POLICY

We evaluate PATCH POLICY and baselines on three real-world manipulation tasks: Cable Insertion, Pen Collection, and Tool Hanging. These tasks are designed to test precise, long-horizon, and contact-rich manipulation, respectively. Detailed task descriptions can be found in Section A.2. We use DINOv2 [6] (ViT-S) patch features for all real-world experiments, a compact yet top-performing backbone in our representation study (Section 3.5) that is well established for strong visual understanding [6, 20]. Additionally, in Section A.3 and Section A.3.1, we evaluate a zero-shot general object pickup policy trained with our method, and compare with the original global-pooled policy following the setup of Contact-Anchored Policies [27].

Table 2 reports cumulative success rates through each task stage. Nearly every method completes the initial grasp; as we progress into the task, however, we find that PATCH POLICY outperforms baselines on every task (visualizations in Fig. 4). The improvement is most significant on the lowest-tolerance task of Cable Insertion, where the barrel jack must be inserted with a $\sim$2 mm tolerance. The improvements over global pooled features mirror our simulation findings. Notably, PATCH POLICY outperforms a fine-tuned OpenVLA-OFT on all three tasks, showing that for in-domain task learning, a lightweight policy trained from scratch over frozen dense features can be competitive with a heavy pretrained VLA fine-tuned on the same data. In Cable Insertion, OpenVLA-OFT often gets stuck near the socket, suggesting that these failures are dominated by fine-grained closed-loop control after contact, rather than task-level semantic understanding.

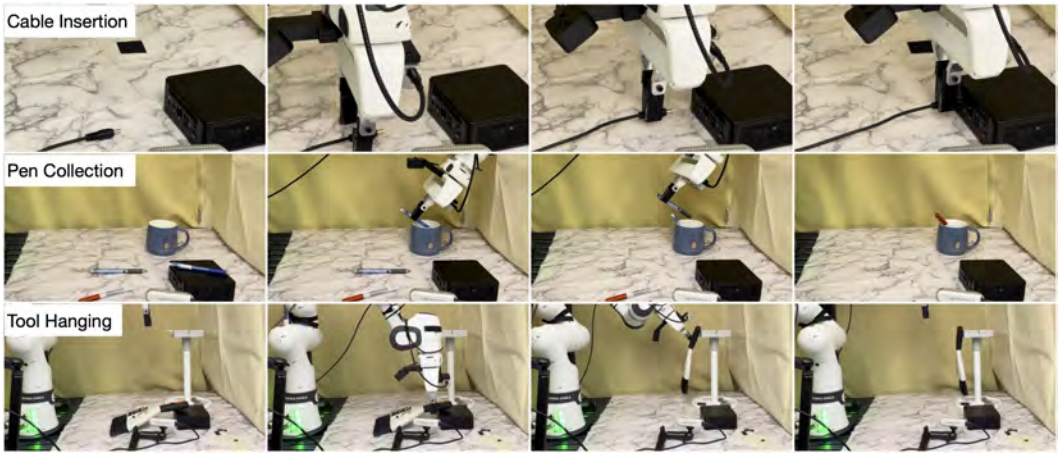

Figure 4: Real-robot rollout examples for the three evaluation tasks: cable insertion, pen collection, and tool hanging. Each row shows one task rollout and visualizes key stages of the policy execution.

3.5 Benchmarking Pre-trained Visual Representations

While PATCH POLICY achieves superior results using WebSSL patch features, it remains to be seen how alternative pre-trained representations impact downstream policy learning. To investigate the extent to which the choice of vision backbone influences performance, we evaluate five state-

Table 2: Real-robot success rates through task stages, 20 trials.

Visual Representation	Policy	Task Stage		
Cable Insertion		Picked up cable	Plugged into port	Fully inserted
DINOv2 Patch	VQ-BeT **(Ours)**	**1.00**	**0.85**	**0.70**
DINOv2 `CLS`	VQ-BeT	**1.00**	0.70	0.60
ResNet-18 Patch	ACT	**1.00**	0.40	0.35
DINOv2 + SigLIP Patch	OpenVLA-OFT	**1.00**	0.55	0.30
Pen Collection		First pen placed	Second pen placed	Third pen placed
DINOv2 Patch	VQ-BeT **(Ours)**	**1.00**	**1.00**	**0.85**
DINOv2 `CLS`	VQ-BeT	**1.00**	0.95	0.65
ResNet-18 Patch	ACT	**1.00**	0.85	0.65
DINOv2 + SigLIP Patch	OpenVLA-OFT	**1.00**	0.85	0.60
Tool Hanging		Picked up	Reached hook	Tool placed
DINOv2 Patch	VQ-BeT **(Ours)**	**1.00**	**0.90**	**0.90**
DINOv2 `CLS`	VQ-BeT	**1.00**	0.75	0.70
ResNet-18 Patch	ACT	**1.00**	0.85	0.85
DINOv2 + SigLIP Patch	OpenVLA-OFT	0.95	**0.90**	0.65

of-the-art visual representations: DINOv2 [6], DINOv3 [11], WebSSL [12], V-JEPA 2 [14], and SigLIP 2 [9]. Detailed descriptions of each encoder are provided in Section A.4.

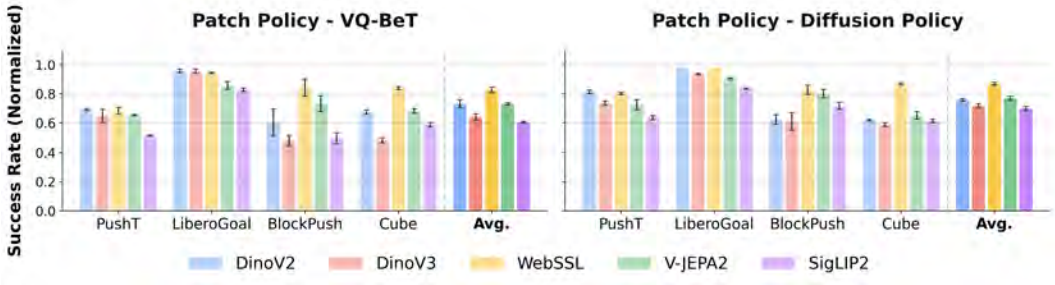

Figure 5: **Comparison of PATCH POLICY across various pretrained visual representations**. We report the mean and standard deviation across runs with three random seeds. Our results suggest that DINOv2 and WebSSL are the most effective vision backbones for robot learning tasks. We normalize BlockPush and Cube results to $[0, 1]$ to facilitate comparison across the entire task suite.

For all experiments, we freeze the image encoder and exclusively train the policy. This approach isolates and evaluates the out-of-the-box representation capabilities of each backbone. This frozen setup also allows us to precompute the visual embeddings, significantly accelerating training.

We report the performance on the simulated environments in Figure 5 and full results in Table 7 and Table 8, as well as the average performance across all tasks per policy backbone. As illustrated in Figure 5, WebSSL and DINOv2 achieve the highest performance across the majority of tasks. This establishes them as robust visual backbones for robotic control and aligns with previous findings in world modeling [28]. SigLIP 2 falls short across the environments. We hypothesize that SigLIP 2's emphasis on semantic language-image alignment sacrifices the dense geometric features necessary for manipulation. Since our tasks prioritize spatial reasoning over linguistic understanding, this vision-language grounding provides a less effective signal for policy learning.

Notably, the relative ranking of visual representations remains remarkably consistent across different policy architectures for any given task. This consistency suggests that the quality of the visual representation is still a primary bottleneck for policy learning, independent of the downstream action head. Furthermore, the average ranking of these representations remains stable across the entire task

suite, underscoring the generalizability of certain pre-training objectives. Based on these findings, we recommend the use of WebSSL or DINOv2 as the vision backbones for robot learning tasks.

3.6 Is PATCH POLICY computationally efficient?

Beyond task performance, we evaluate the computational feasibility of PATCH POLICY for real-time robotic applications. In Table 3, we compare the parameter counts and inference latency (ms) of PATCH POLICY variants against baseline policies on an NVIDIA H200 GPU.

Table 3: Computational Resources and Inference Speed of PATCH POLICY.

Method	Total Params	Trainable Params	Inference Latency (ms)
VQ-BeT (ResNet-18)	39.95M	28.77M	5.79
Ours - VQ-BeT (DINOv2)	51.55M	29.49M	10.99
Ours - VQ-BeT (WebSSL)	334.00M	30.34M	21.43
DP (ResNet-18)	29.35M	9.09M	421.89
Ours - DP (DINOv2)	40.43M	9.19M	445.85
Ours - DP (WebSSL)	303.66M	9.35M	451.68
OpenVLA-OFT	7.61B	177.90M	61.71
ACT	83.85M	83.85M	8.63

Parameter Efficiency. PATCH POLICY beats OpenVLA-OFT with under 5% of its parameters with ViT-L (as little as $\sim$0.7% with ViT-S), showing that what matters is not scale but direct access to dense patch features from a frozen, Internet-pretrained backbone. Section A.5 has training details and model size ablations.

Inference Latency. Real-time control typically requires high-frequency feedback loops. Here we report the raw model inference latency, defined as the time required for a single forward pass which does not include temporal speedups from action chunking, as such techniques can be applied orthogonally to all evaluated methods. All evaluations are on a single H200 GPU. Our VQ-BeT variants demonstrate exceptional speed even when processing dense DINOv2 patch features (10.99 ms), comparable to ACT with ResNet-18 at 8.63 ms. In contrast, while OpenVLA-OFT is optimized for efficiency, it still requires 61.71 ms per forward pass. We observe that switching from global representations (ResNet) to patch features (DINOv2) in PATCH POLICY results in a predictable increase in latency due to the longer token sequence (5.79 ms $\rightarrow$ 10.99 ms for VQ-BeT); however, the resulting latency remains well within the requirements for high-speed manipulation. Notably, the bottleneck for Diffusion Policy variants remains the iterative denoising process rather than the visual representation, as evidenced by the negligible difference in inference latency between the global (421.89 ms) and patch-based (445.85 ms) DP variants.

Training Cost. PATCH POLICY with DINOv2 patch features converges in 6.5 hours on 1xL40S (6.5 GPU-hours); OpenVLA-OFT converges in 4 hours on 4x L40S (16 GPU-hours); ACT converges in 12 hours on 2xL40S (24 GPU-hours). This efficiency reflects both the small trainable parameter count and the frozen pretrained visual backbone.

3.7 Should we compress patch features?

PATCH POLICY achieves strong performance using raw patch features. A natural question is whether patch tokens for each frame can be compressed to improve efficiency without sacrificing policy performance. We introduce a lightweight convolutional encoder, trained end-to-end with the policy, to compress the frozen DINOv2 patch features spatially, reducing the number of patch tokens per frame. On Push-T (Table 4), spatially downsampling the features results in a significant decrease in task success. This trend confirms that the fine-grained spatial density of the origi-

Table 4: Patch Compression

Resolution	Push-T
256 patches	**0.69**
64 patches	0.52
16 patches	0.53
4 patches	0.51
1 patch	0.48

nal tokens is crucial for precise control. While compression remains a viable trade-off for strictly hardware-constrained deployments, we recommend preserving the uncompressed patch resolution whenever compute permits.

3.8 Additional Ablations

Section A.6 contains more experiments and details on attention masks, model size, and visualizations.

4 Related Work

4.1 Imitation Learning

Imitation Learning (IL) enables agents to learn skills from expert demonstrations without explicit reward engineering [29]. While early IL focused on low-dimensional states, recent paradigms shift toward vision-based policies operating on high-dimensional inputs. Despite advances in expressive architectures and generative objectives that model multimodal actions from large datasets [22, 24, 21, 30, 31, 32, 33, 34, 35], a performance gap persists between state-based and vision-based agents. PATCH POLICY addresses this within vision-based behavioral cloning. Rather than proposing a novel training objective, we close this gap by equipping standard architectures with the dense visual representations well-established in computer vision.

4.2 Visual Representation for Embodied Learning

Visual representation for control has rapidly evolved from in-domain self-supervised methods [36, 37, 23] to Vision Transformers (ViTs) [1]—such as DINO [38, 6, 11], V-JEPA [13, 14], and SigLIP [26, 9]—that extract dense patch features instead of compressed global vectors [39]. Despite their success in world modeling [20, 28, 14], their adoption in robotic learning remains limited. Because standard continuous control backbones assume compressed global states [32, 30, 22, 21], most architectures still rely on global pooling. Aside from computationally heavy VLA models [40], efficient robotic agents rarely process uncompressed spatial tokens. PATCH POLICY bridges this gap by directly integrating foundational patch features into lightweight policies.

4.3 Transformer-based Policies and VLAs

Transformers [19] have achieved remarkable success across NLP [41, 42] and vision [1, 7, 14]. Its capacity for long-range dependency modeling makes it natively suitable for policy learning [43]. Transformer-based policies have been widely adopted in both RL [44, 45] and IL [30, 31, 24, 46, 47, 21]. These models typically assume a compressed global feature vector for vision, discarding spatial granularity. While some agents do utilize multiple spatial tokens [24, 48, 49, 50, 51, 52, 53], they often rely on end-to-end encoder training or finetuning in-domain instead of using pretrained vision backbones. We include ACT [24] as a representative baseline from this category, which operates on patch tokens from a ResNet-18 encoder trained from scratch. One prominent class of methods that utilizes dense features is the Vision-Language-Action (VLA) model, which jointly learns language understanding, control, and perception [40, 34, 54, 55, 56, 57] by fine-tuning large Vision-Language Models (VLM) on large-scale embodied data. The large backbone introduces latency for real-time control, which recent optimizations such as OpenVLA-OFT [25] begin to address. We adopt OpenVLA-OFT as our primary VLA baseline. PATCH POLICY disentangles the utilization of dense visual representations from general language grounding and massive model scale, and provides a lightweight pipeline that directly ingests these dense features for imitation learning.

5 Limitations

While PATCH POLICY effectively leverages dense spatial features for control, several directions remain for future work. First, we focused exclusively on frozen vision backbones, and future work

could explore end-to-end fine-tuning to adapt these representations for specialized visual domains. Second, dense tokens increase sequence length and training time. Optimizations like FlashAttention [58] could accelerate both training and inference. Finally, PATCH POLICY is currently evaluated purely as a behavior cloning policy. Extending this patch-based architecture to reinforcement learning could be a promising direction to surpass the performance ceiling of static expert demonstrations.

6 Conclusion

In this work, we investigate the efficacy of patch-level features for robot learning and introduce PATCH POLICY, an efficient policy class that harnesses pre-trained dense visual representations. It achieves superior performance while remaining computationally lean across parameter count, training compute, and inference latency, outperforming both global-feature policies and heavyweight VLAs. By keeping the visual backbone frozen and ingesting its patch tokens directly through a block-causal attention mask, PATCH POLICY provides the robotics community with a lightweight, drop-in pipeline to readily absorb continuing progress in visual representation learning, without sacrificing the fast training and high-frequency control that real-world manipulation demands.

Acknowledgments

We would like to thank Mahi Shafiullah, Irmak Guzey, Kevin Wu, Hengkai Pan, Zifan Zhao, Alex Xiaole Jiang, Andre Wang, Zicheng Teng, Kanad Patel, Yvonne Wu, Zavier Andrianarivo for their valuable discussion and feedback. This work was supported by grants from LG, Qualcomm, Honda, Microsoft, Hyundai, NSF award 2339096 and ONR awards N00014-21-1-2758 and N00014-22-1-2773, and AFOSR under grant FA95502310139. Lerrel Pinto is supported by the Sloan, Packard, and CIFAR Fellowships.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. URL https://api.semanticscholar.org/CorpusID:225039882.

[2] K. He, X. Chen, S. Xie, Y. Li, P. Doll'ar, and R. B. Girshick. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2021. URL https://api.semanticscholar.org/CorpusID:243985980.

[3] M. Caron, H. Touvron, I. Misra, H. J'egou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. URL https://api.semanticscholar.org/CorpusID:233444273.

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231591445.

[5] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9620–9629, 2021. URL https://api.semanticscholar.org/CorpusID:233024948.

[6] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. B. Huang, S.-W. Li, I. Misra, M. G. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal,

P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision. *ArXiv*, abs/2304.07193, 2023. URL https://api.semanticscholar.org/CorpusID:258170077.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. URL https://api.semanticscholar.org/CorpusID:232352874.

[8] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[9] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. M. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, O. H'enaff, J. Harmsen, A. Steiner, and X.-Q. Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *ArXiv*, abs/2502.14786, 2025. URL https://api.semanticscholar.org/CorpusID:276482166.

[10] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Surís, C. K. Ryali, K. V. Alwala, H. Khedr, A. Huang, J. Lei, T. Ma, B. Guo, A. Kalla, M. Marks, J. Greer, M. Wang, P. Sun, R. Radle, T. Afouras, E. Mavroudi, K. Xu, T.-H. Wu, Y. Zhou, L. Momeni, R. Hazra, S. Ding, S. Vaze, F. Porcher, F. Li, S. Li, A. Kamath, H. K. Cheng, P. Doll'ar, N. Ravi, K. Saenko, P. Zhang, and C. Feichtenhofer. Sam 3: Segment anything with concepts. *ArXiv*, abs/2511.16719, 2025. URL https://api.semanticscholar.org/CorpusID:283226295.

[11] O. Sim'eoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. J'egou, P. Labatut, and P. Bojanowski. Dinov3. 2025. URL https://api.semanticscholar.org/CorpusID:280649654.

[12] D. Fan, S. Tong, J. Zhu, K. Sinha, Z. Liu, X. Chen, M. Rabbat, N. Ballas, Y. LeCun, A. Bar, and S. Xie. Scaling language-free visual representation learning. *ArXiv*, abs/2504.01017, 2025. URL https://api.semanticscholar.org/CorpusID:277467353.

[13] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL https://arxiv.org/abs/2404.08471.

[14] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[18] S. Dasari, M. K. Srirama, U. Jain, and A. Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*. PMLR, 2023.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

[20] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. DINO-WM: world models on pre-trained visual features enable zero-shot planning. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

[21] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. Muhammad, M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *ArXiv*, abs/2403.03181, 2024. URL https://api.semanticscholar.org/CorpusID:268248763.

[22] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.026. URL https://doi.org/10.15607/RSS.2023.XIX.026.

[23] Z. J. Cui, H. Pan, A. Iyer, S. Haldar, and L. Pinto. Dynamo: In-domain dynamics pretraining for visuo-motor control. *ArXiv*, abs/2409.12192, 2024. URL https://api.semanticscholar.org/CorpusID:272709175.

[24] T. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *ArXiv*, abs/2304.13705, 2023. URL https://api.semanticscholar.org/CorpusID:258331658.

[25] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *ArXiv*, abs/2502.19645, 2025. URL https://api.semanticscholar.org/CorpusID:276647709.

[26] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.

[27] Z. J. Cui, O. Rayyan, H. Etukuru, B. Tan, Z. Andrianarivo, Z. Teng, Y. Zhou, K. Mehta, N. Wojno, K. Y. Wu, M. H. Anjaria, Z. Wu, M. Mao, G. Zhang, B. Shah, Y. Kim, S. Chintala, L. Pinto, and N. M. M. Shafiullah. Contact-anchored policies: Contact conditioning creates strong robot utility models. *arXiv preprint arXiv:2602.09017*, 2026.

[28] R. G. Goswami, A. Bar, D. Fan, T.-Y. Yang, G. Zhou, P. Krishnamurthy, M. Rabbat, F. Khorrami, and Y. LeCun. World models can leverage human videos for dexterous manipulation. *ArXiv*, abs/2512.13644, 2025. URL https://api.semanticscholar.org/CorpusID:283896258.

[29] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[30] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. *ArXiv*, abs/2206.11251, 2022. URL https://api.semanticscholar.org/CorpusID:249926747.

[31] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *ArXiv*, abs/2210.10047, 2022. URL https://api.semanticscholar.org/CorpusID:252968170.

[32] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

[33] K. Rana, R. Lee, D. Pershouse, and N. Suenderhauf. Imle policy: Fast and sample efficient visuomotor policy learning via implicit maximum likelihood estimation. *arXiv preprint arXiv:2502.12371*, 2025.

[34] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[35] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[36] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2019. URL https://api.semanticscholar.org/CorpusID:207930212.

[37] J.-B. Grill, F. Strub, F. Altch'e, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733, 2020. URL https://api.semanticscholar.org/CorpusID:219687798.

[38] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[39] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3M: A universal visual representation for robot manipulation. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 892–909. PMLR, 2022. URL https://proceedings.mlr.press/v205/nair23a.html.

[40] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, K. Choromanski, T. Ding, D. Driess, K. A. Dubey, C. Finn, P. R. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, I. Leal, S. Levine, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. S. Ryoo, G. Salazar, P. R. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. H. Vuong, A. Wahid, S. Welker, P. Wohlhart, T. Xiao, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *ArXiv*, abs/2307.15818, 2023. URL https://api.semanticscholar.org/CorpusID:260293142.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

[42] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[43] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General perception with iterative attention. *ArXiv*, abs/2103.03206, 2021. URL https://api.semanticscholar.org/CorpusID:232110866.

[44] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235294299.

[45] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235313679.

[46] Y. Su, X. Zhan, H. Fang, H. Xue, H.-S. Fang, Y.-L. Li, C. Lu, and L. Yang. Dense policy: Bidirectional autoregressive learning of actions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14486–14495, 2025.

[47] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *ArXiv*, abs/2406.07539, 2024. URL https://api.semanticscholar.org/CorpusID:270379931.

[48] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. A. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *ArXiv*, abs/2212.06817, 2022. URL https://api.semanticscholar.org/CorpusID:254591260.

[49] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. D. Edwards, N. M. O. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent. *ArXiv*, abs/2205.06175, 2022. URL https://api.semanticscholar.org/CorpusID:248722148.

[50] Z. Hou, T. Zhang, Y. Xiong, H. Pu, C. Zhao, R. Tong, Y. Qiao, J. Dai, and Y. Chen. Diffusion transformer policy. *arXiv preprint arXiv:2410.15959*, 2024.

[51] O. M. Team, D. Ghosh, H. R. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, P. R. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. *ArXiv*, abs/2405.12213, 2024. URL https://api.semanticscholar.org/CorpusID:266379116.

[52] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[53] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[54] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. H. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. R. Florence. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:257364842.

[55] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. $\pi 0$: A vision-language-action flow model for general robot control. *ArXiv*, abs/2410.24164, 2024. URL https://api.semanticscholar.org/CorpusID:273811174.

[56] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. R. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. π0.5: a vision-language-action model with open-world generalization. *ArXiv*, abs/2504.16054, 2025. URL https://api.semanticscholar.org/CorpusID:277993634.

[57] P. Intelligence, A. A. Amin, R. J. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. π*0.6: a vla that learns from experience. *ArXiv*, abs/2511.14759, 2025. URL https://api.semanticscholar.org/CorpusID:283080952.

[58] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

[59] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[60] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. *ArXiv*, abs/2410.20092, 2024. URL https://api.semanticscholar.org/CorpusID:273654871.

[61] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

A Appendix

A.1 Implementation and Baselines

Our implementation and baselines are built upon the following open-source repositories:

1. DynaMo: https://github.com/jeffacce/dynamo_ssl
2. VQ-BeT: https://github.com/jayLEE0301/vq_bet_official
3. Diffusion Policy: https://github.com/real-stanford/diffusion_policy
4. OpenVLA-OFT: https://github.com/moojink/openvla-oft
5. ACT: https://github.com/tonyzhaozh/act

A.2 Environments and Tasks

We evaluate PATCH POLICY across four simulated environments (Push-T, LIBERO Goal, Block-Push, Cube) with 2D-to-7D action spaces, and three real-world tasks using a 7-DoF Franka arm with a parallel-jaw gripper. The real-world tasks comprise: (1) *Tool Hanging*, (2) *Pen Collection* (long-horizon, small objects), (3) *Cable Insertion* (low-tolerance insertion). We additionally test the generalization capabilities of PATCH POLICY in another real robot task: *Franka Pickup* (unseen object generalization). For each environment, we detail the experimental setup, dataset size and collection process, and the metrics for evaluation. Refer to Figure 3 for environment visualizations.

1. **Push-T [22]:** This environment requires a pusher agent with 2-dimensional action space to maneuver a green T-shaped block into a fixed, gray T-shaped target zone. Observations consist of a shape 224×224 top-down image of the scene. The dataset has 206 expert trajectories from Diffusion Policy [22]. We evaluate over 100 rollouts and report the final spatial coverage proportion of the green block over the target (maximum 1.0). See Figure 11 for a sample rollout of PATCH POLICY on Push-T.
2. **LIBERO Goal [59]:** This environment suite from the LIBERO benchmark consists of 10 manipulation tasks in one scene with a 7-DoF Franka arm.

   Observations are multi-view, providing shape 224×224 images from both a third-person and a wrist-mounted camera. The dataset contains 50 expert trajectories per task, yielding 500 total demonstrations. We evaluate the suite across 100 trials (10 per goal) and report the overall success rate (maximum 1.0). See Figure 11 for a sample rollout of PATCH POLICY for each task.
3. **BlockPush [32]:** This environment features two blocks and two square target zones. A robotic pusher, controlled via 2D end-effector translation, must push both blocks into designated targets (either same- or opposite-colored). The state is captured via two 224×224 third-person camera views. We train on 1000 expert demonstrations and evaluate across 100 rollouts, reporting the mean number of correctly placed blocks (maximum 2.0).
4. **Cube [60]:** This environment from OGBench [60] features a 6-DoF UR5e robot arm interacting with two cube blocks in a pick-and-place task. The goal is for the arm to stack cubes into a fixed designated configuration at the center of the plane. The observation is a 224×224 image of a third-person view of the scene. The initial positions of the cube blocks are randomized. For training, we generate a dataset of 550 successful trajectories, where success requires correct stacking order and placement within 200 steps. The dataset was collected by an expert policy with temporally correlated noise following OGBench [60]. We evaluate 100 rollouts and report the mean number of blocks reaching target locations (maximum 2.0). See Figure 11 for a sample rollout of PATCH POLICY for each task.
5. **Tool Hanging:** In the tool hanging task, a brush is initially placed on the table. The robot must pick up the brush and hang it on a small hook mounted on a stand. A trial is considered successful if the robot hangs the brush stably on the hook. This task requires grasping

the tool from the table, accurately positioning it relative to a small hook, and achieving a stable final configuration upon release. We use two wrist-mounted cameras, one facing forward and one facing downward. We collect 50 demonstration episodes, corresponding to approximately 18 minutes of real-robot data.

6. **Pen Collection:** In the pen collection task, the robot must pick up three pens and place all of them into a pen holder. Each trial contains three different pens and three possible initial positions. The pens are not assigned to fixed positions and can be permuted across trials. The initial pose of each pen is randomized within approximately ± 1 cm in position and $\pm 15^{\circ}$ in orientation. This task evaluates multi-object manipulation, robustness to object-pose variation, and long-horizon execution over repeated pick-and-place stages. We use two wrist-mounted cameras, one facing forward and one facing downward, to provide complementary visual observations during grasping and placement. We collect 52 demonstration episodes, corresponding to approximately 45 minutes of real-robot data.
7. **Cable Insertion:** In the cable insertion task, the robot is required to pick up a power cable and insert it into a NUC. The NUC position is fixed across all demonstrations and evaluation trials, while the initial pose of the cable is randomized within a 2cm $\times$ 2cm square. The final insertion tolerance is around 2mm. This task requires the robot to first grasp a barrel jack, align the connector with the port, and complete a precise insertion. To ensure that the insertion process is fully observable, we use two camera views: a wrist-mounted camera and a side-view camera. We collect 101 demonstration episodes, corresponding to approximately 37 minutes of real-robot data.
8. **CAP - EgoGym [27]:** Introduced in Contact-Anchored Policies (CAP) [27], this simulated benchmark suite consists of three zero-shot generalization environments: general object pickup, door/drawer opening, and door/drawer closing. The dataset contains 14,606 real-world demonstrations for pickup, 3,690 for opening, and 2,069 for closing, collected via a handheld tool. Figure 13 shows sample trajectories from the dataset for each of the tasks. The simulation procedurally generates scenes with diverse textures to test zero-shot generalization. Figure 12 shows PATCH POLICY rollouts in EgoGym.
9. **CAP - Franka Pickup:** For real world evaluations, we set up a physical Franka FR3 robot and ten evaluation objects exactly following the hardware and object configuration established in CAP [27]. The observation is a 224×224 image from a wrist-mounted camera, and a 3D point specifying the expected physical contact location. For each evaluation, we evaluate with five objects in scene, and condition the policy to pick up each of the five objects for ten trials, for the two sets of five objects, for a total of 100 evaluations. See Figure 9 and Figure 10 for rollout trajectories of PATCH POLICY.

A.3 Zero-shot Manipulation in the Real World

To evaluate whether gains from PATCH POLICY apply in the zero-shot generalization domain, where the policy is trained on a large-scale real-world dataset, and evaluated in out-of-distribution environments and on novel objects (Figure 6), we train a VQ-BeT policy on the CAP object pickup dataset with DINOv2 patch features. We evaluate it on the real Franka robot environment, where the robot attempts to pick up 10 objects unseen during training for a total of 100 trials (Section A.2). We compare it with the original CAP checkpoint with global-pooled features on the CAP gripper embodiment, following the CAP evaluation criteria (Section 3.2). We see that PATCH POLICY improves over the vanilla CAP checkpoint in zero-shot object pickup.

Table 5: Real-world zero-shot object pickup results.

Method	Real Franka Pickup
CAP	79%
Ours	**87%**

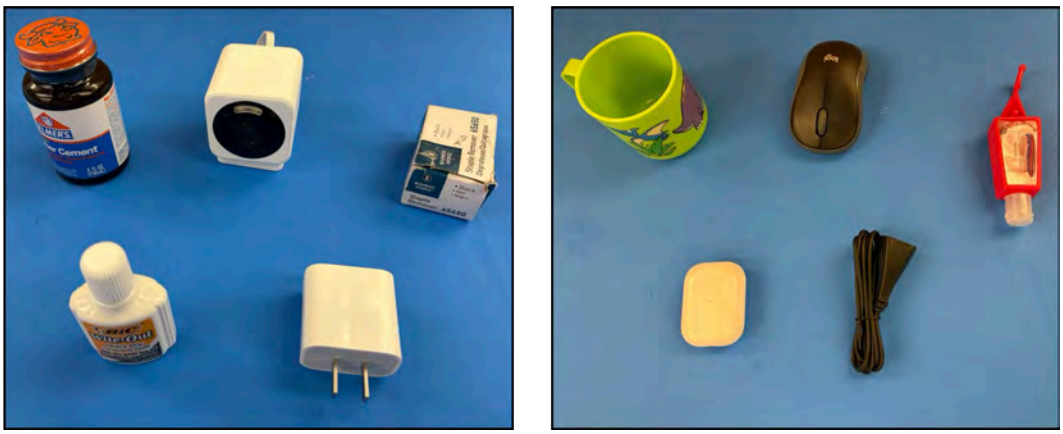

Figure 6: We evaluate on 10 unseen objects for Real Franka Pickup, following the evaluation criteria in CAP [27].

Figure 9 shows rollout success trajectories for each object, and Figure 10 shows rollout failure modes (early grasping, and overshooting).

A.3.1 Zero-shot real-to-sim evaluations.

Additionally, CAP [27] and SIMPLER [61] have shown that for policies trained on real-world datasets, simulation evaluations can be highly predictive of real-world performance. To this end, we train patch-based policies on object pickup, door/drawer opening, and closing data, and compare them with the released global feature CAP checkpoints, evaluating both on diverse background textures and objects for 5000 episodes in EgoGym. We see in Table 6 that patch policies outperform global feature policies across all three tasks in the EgoGym benchmark in diverse evaluations. Rollout trajectory samples are shown in Figure 12.

Table 6: Real-to-sim evaluations on EgoGym.

Method	EgoGym Pickup	EgoGym Open	EgoGym Close
CAP	75.78%	67.88%	86.50%
Ours	**79.50%**	**71.40%**	**92.44%**

A.4 Benchmarking Pretrained Visual Representations for Policy Learning

While PATCH POLICY achieves superior results using DINOv2 patch features, it remains to be seen how alternative pre-trained representations impact downstream policy learning. To investigate the extent to which the choice of vision backbone influences performance, we evaluate PATCH POLICY across the following state-of-the-art visual representations:

1. **DINOv2 [6]:** A Vision Transformer pre-trained on a curated dataset of 142M images (`LVD-142M`) using discriminative and reconstructive self-supervised losses; for a 224×224 input, we use the ViT-S/14 backbone yielding $16 \times 16 \times 384$ patch features.
2. **DINOv3 [11]:** An evolution of the DINO framework scaling to larger, internet-scale datasets with improved stability; for a 224×224 input, we employ the ViT-S/16+ variant providing $14 \times 14 \times 384$ patch features.
3. **WebSSL [12]:** A model that applies the DINO self-supervised loss to massive, uncurated web-crawled datasets; for a 224×224 input, the encoder outputs $16 \times 16 \times 1024$ patch features.

4. **V-JEPA 2 [14]:** A Video Joint-Embedding Predictive Architecture trained on 1M+ hours of videos to learn temporal and physical dynamics; for a 224×224 input, the encoder outputs $14 \times 14 \times 1024$ patch embeddings.
5. **SigLIP 2 [9]:** An improved version of the Sigmoid Language-Image Pre-training model trained on the WebLI dataset (internet-scale image-text pairs); for a 224×224 input, the encoder outputs $14 \times 14 \times 768$ patch features.

In this section, we present the comprehensive numerical results for benchmarking PATCH POLICY across the various pre-trained visual representations illustrated in Figure 5. For each configuration, we performed evaluations over three independent seeds, reporting the resulting mean and standard deviations. The performance for PATCH POLICY-VQ-BeT is detailed in Table 7, while the corresponding results for PATCH POLICY-Diffusion Policy are provided in Table 8.

Table 7: Effect of Pre-trained Visual Representations on PATCH POLICY - VQ-BeT. Values within 2% of the maximum performance for each task are shown in bold.

Method	Push-T	LIBERO Goal	BlockPush	Cube
Ours – DINOv2	**0.69** ±0.01	**0.96** ±0.01	1.20 ±0.23	1.35 ±0.03
Ours – DINOv3	0.65 ±0.05	**0.95** ±0.02	0.96 ±0.09	0.96 ±0.04
Ours – WebSSL	**0.68** ±0.02	**0.94** ±0.01	**1.68** ±0.15	**1.68** ±0.02
Ours – V-JEPA2	0.65 ±0.01	0.86 ±0.03	1.46 ±0.13	1.36 ±0.03
Ours – SigLIP2	0.51 ±0.01	0.83 ±0.01	0.99 ±0.10	1.17 ±0.03

Table 8: Effect of Pre-trained Visual Representations on PATCH POLICY - Diffusion Policy. Values within 2% of the maximum performance for each task are shown in bold.

Method	Push-T	LIBERO Goal	BlockPush	Cube
Ours – DINOv2	**0.81** ±0.01	**0.98** ±0.00	1.25 ±0.08	1.24 ±0.01
Ours – DINOv3	0.73 ±0.02	0.94 ±0.01	1.22 ±0.15	1.17 ±0.03
Ours – WebSSL	**0.80** ±0.01	**0.98** ±0.00	**1.65** ±0.08	**1.73** ±0.02
Ours – V-JEPA2	0.72 ±0.04	0.91 ±0.01	1.60 ±0.07	1.30 ±0.05
Ours – SigLIP2	0.64 ±0.02	0.83 ±0.01	1.43 ±0.06	1.23 ±0.03

A.5 Hyperparameters

We present the hyperparameters for PATCH POLICY training and relevant implementation details below. All hyperparameters of the PATCH POLICY-VQ-BeT variant and PATCH POLICY-Diffusion Policy variant have been reported in Table 9 and Table 11. We further ablate PATCH POLICY's model size in Section A.6.2.

A.6 Additional ablations

A.6.1 PATCH POLICY Attention Mask Ablation

Transformer-based policies traditionally condition on a single token per observation step, allowing direct application of standard token-level causal attention. In PATCH POLICY, each observation is now a sequence of tokens and we thus apply a block-causal attention mask as mentioned in Section 2.2. Under this setting, every patch token attends fully to all other patch tokens within the same frame, while attention across frames remains strictly causal. In this section, we ablate this block-causal attention along with two other masking choices: 1) full attention, with no masking; 2) token-causal, the naive implementation that treats every patch as an independent timestep.

As shown in Table 12, block-causal matches or exceeds the alternatives on most (policy head, task) pair, with the largest gains on Diffusion Policy. Full attention lets every position attend to patches

Table 9: PATCH POLICY - VQ-BeT Hyperparameters

Hyperparameter	Push-T	LIBERO Goal	BlockPush	Cube	Pickup	Open	Close
VQ-VAE Configuration							
Latent Dimension	512	512	512	512	512	512	512
Codebook Size	16	16	16	16	16	16	16
Number of Groups	2	2	2	2	2	2	2
Architecture							
Number of Layers (N)	8	6	8	8	8	8	8
Number of Heads (n_{heads})	8	6	8	8	8	8	8
Embedding Dimension (d_{emb})	512	120	512	512	512	512	512
Policy							
Window Size	5	10	3	5	3	3	3
Action Window	5	1	1	5	1	1	1
Training							
Learning Rate	5×10^{-5}	5×10^{-5}	1×10^{-4}	5×10^{-5}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Weight Decay	2×10^{-4}	2×10^{-4}	0	2×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Batch Size	8	32	32	128	256	256	256

Table 10: PATCH POLICY - Real Robot VQ-BeT Hyperparameters

Hyperparameter	Cable Insertion	Pen Collection	Tool Hanging
VQ-VAE Configuration			
Latent Dimension	512	512	512
Codebook Size	16	16	16
Number of Groups	2	2	2
Architecture			
Number of Layers (N)	8	8	8
Number of Heads (n_{heads})	8	8	8
Embedding Dimension (d_{emb})	384	384	384
Policy			
Window Size	2	2	2
Action Window	10	5	5
Training			
Learning Rate	3×10^{-4}	3×10^{-4}	3×10^{-4}
Weight Decay	1×10^{-4}	1×10^{-4}	1×10^{-4}
Batch Size	128	128	128

from future frames during training, thus violating the structure of the underlying task, mildly degrading performance. Token-causal preserves causality but slices the conditioning within a frame. The downstream effect depends on where in the sequence the policy's loss is applied. For VQ-BeT, predictions are read out from the last patch of each frame, so under token-causal it already attends to the same context as block-causal. Therefore, the two are nearly tied. For Diffusion Policy, the denoising loss is applied at every decoder position: early positions see only a partial fraction of the first frame's patches, causing significant performance drop on Cube.

A.6.2 PATCH POLICY Model Size Ablation

In this section, we investigate how PATCH POLICY's performance scales with model size for both the VQ-BeT and Diffusion Policy variants. We sweep for different number of layers (N), number of heads (n_{heads}), and embedding dimension (d_{emb}) of the transformer policy backbone. As detailed in Table 13, performance consistently scales with increased model capacity across both policy variants.

A.7 Additional figures and details

Figure 7 shows representative successful rollout trajectories of PATCH POLICY-VQ-BeT for the three real-world manipulation tasks: Cable Insertion, Pen Collection, and Tool Hanging. Figure 8 shows representative failure rollouts for the same tasks.

Table 11: PATCH POLICY - Diffusion Policy Hyperparameters

Hyperparameter	**Push-T**	**LIBERO Goal**	**BlockPush**	**Cube**
Architecture				
Number of Layers (N)	8	8	8	8
Number of Heads (n_{heads})	4	4	8	4
Embedding Dim (d_{emb})	256	256	512	256
Policy				
Observation Horizon	2	2	3	2
Action Horizon	5	3	1	3
Training				
Learning Rate	1×10^{-4}	5×10^{-5}	1×10^{-4}	1×10^{-4}
Weight Decay	0	2×10^{-4}	0	0
Batch Size	256	256	256	256

Table 12: Attention mask ablation for PATCH POLICY. Block-causal matches or outperforms alternative masking choices across two policy heads on two environments.

	Ours - VQ-BeT		Ours - DP	
Mask	Push-T	Cube	Push-T	Cube
Full	0.64	1.09	**0.83**	1.10
Token-causal	**0.73**	1.36	0.70	0.11
Block-causal (ours)	0.70	**1.38**	**0.83**	**1.24**

Figure 9 shows rollout success trajectories for each object in the Real Franka object pickup experiment. Figure 10 shows rollout failure modes (early grasping, and overshooting) in the Real Franka object pickup experiment.

Figure 11 shows PATCH POLICY-VQ-BeT rollout trajectories for Push-T, Cube, and LIBERO Goal.

Figure 12 shows PATCH POLICY-VQ-BeT rollout trajectories for object pickup, door opening, and closing tasks in EgoGym.

Figure 13 shows sample trajectories of object pickup, opening, and closing from the CAP dataset.

Table 13: Model Size Ablation: Architectural Parameters and Performance on Push-T for PATCH POLICY with DINOv2 (ViT-S) patch features. We report the final mean coverage of the T-shaped block over the target T across 100 evaluation rollouts.

Method	N	n_{heads}	d_{emb}	Size	Final Coverage (↑)
Ours – VQ-BeT	4	4	64	25.62M	0.50
Ours – VQ-BeT	6	6	120	26.64M	0.57
Ours – VQ-BeT	8	8	512	51.55M	0.69
Ours – Diffusion Policy	4	4	64	22.77M	0.07
Ours – Diffusion Policy	6	6	120	25.30M	0.56
Ours – Diffusion Policy	8	4	256	40.43M	0.83

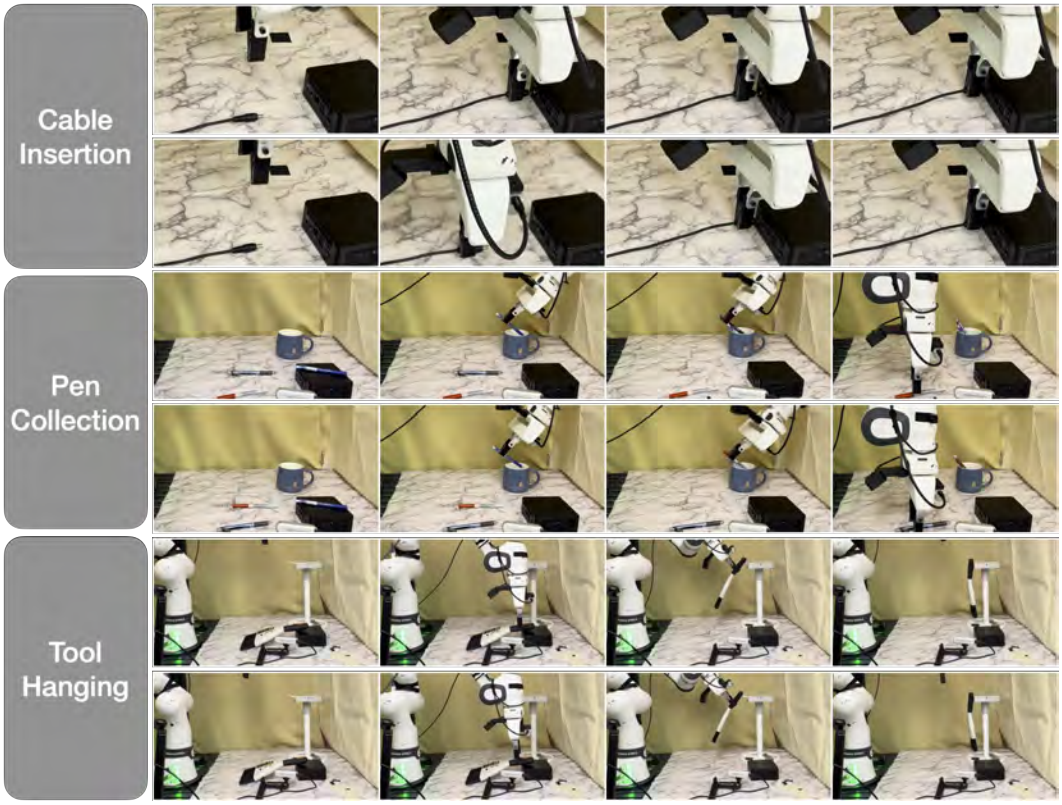

Figure 7: Successful rollouts of PATCH POLICY for Cable Insertion, Pen Collection, and Tool Hanging.

Figure 8: Failure rollouts of PATCH POLICY for Cable Insertion, Pen Collection, and Tool Hanging.

Figure 9: CAP real Franka object pickup successes.

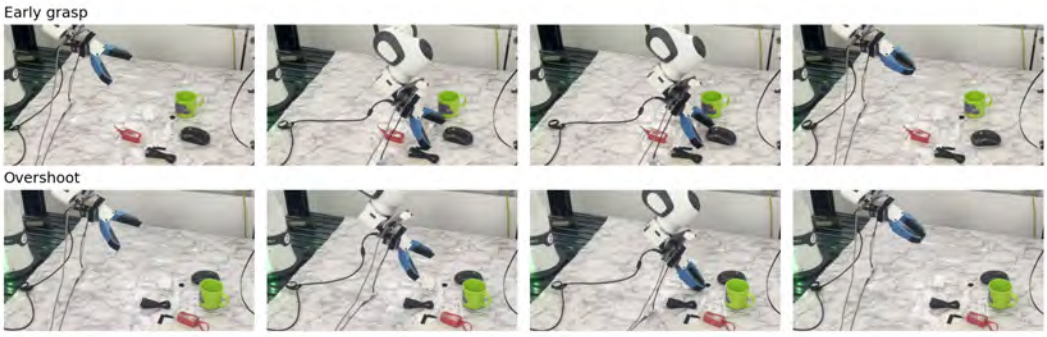

Figure 10: CAP real Franka object pickup failure modes.

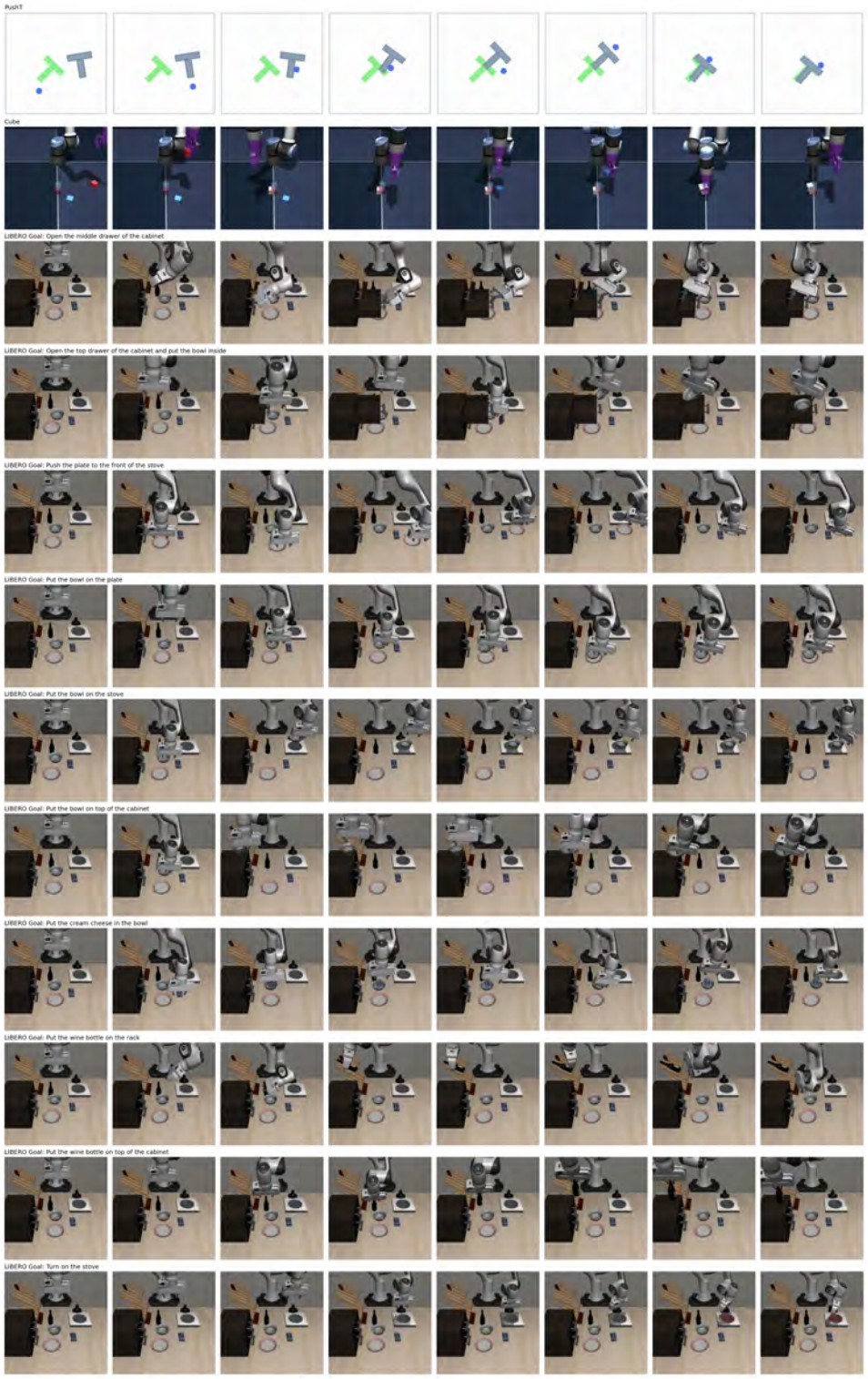

Figure 11: Push-T, Cube, and LIBERO Goal environment Ours - VQ-BeT evaluation rollouts.

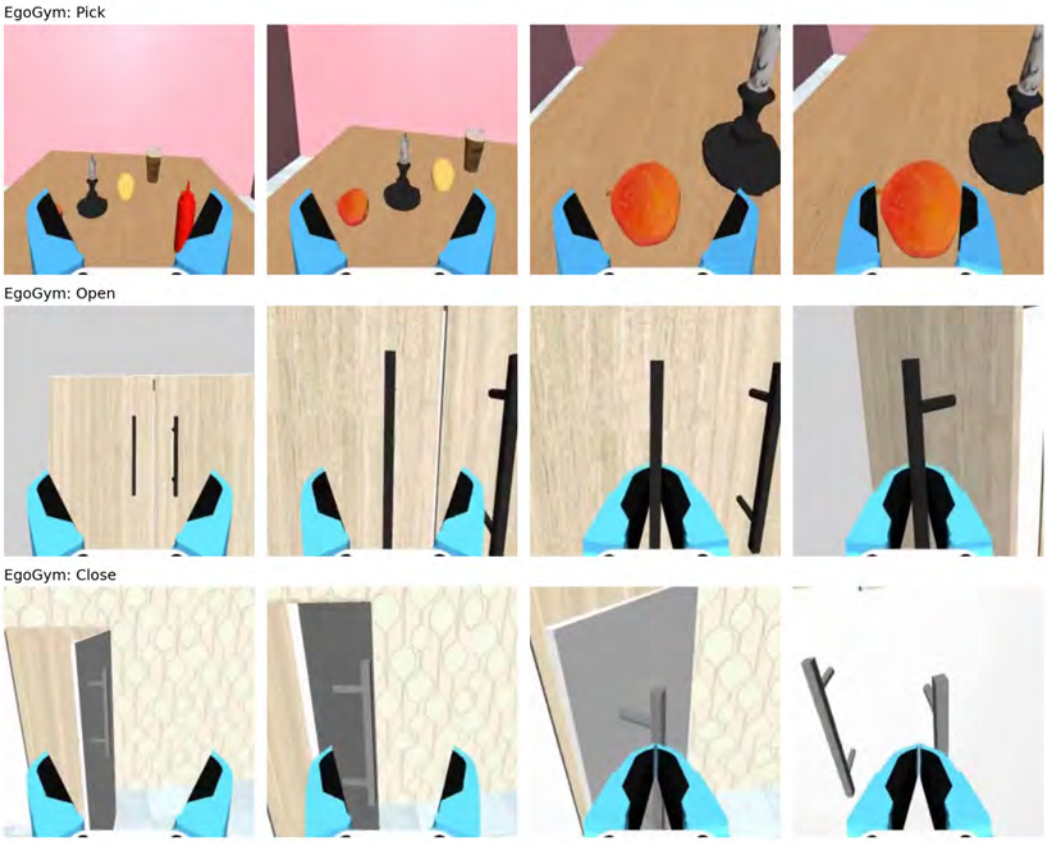

Figure 12: EgoGym pick, open, and close PATCH POLICY evaluation rollouts.

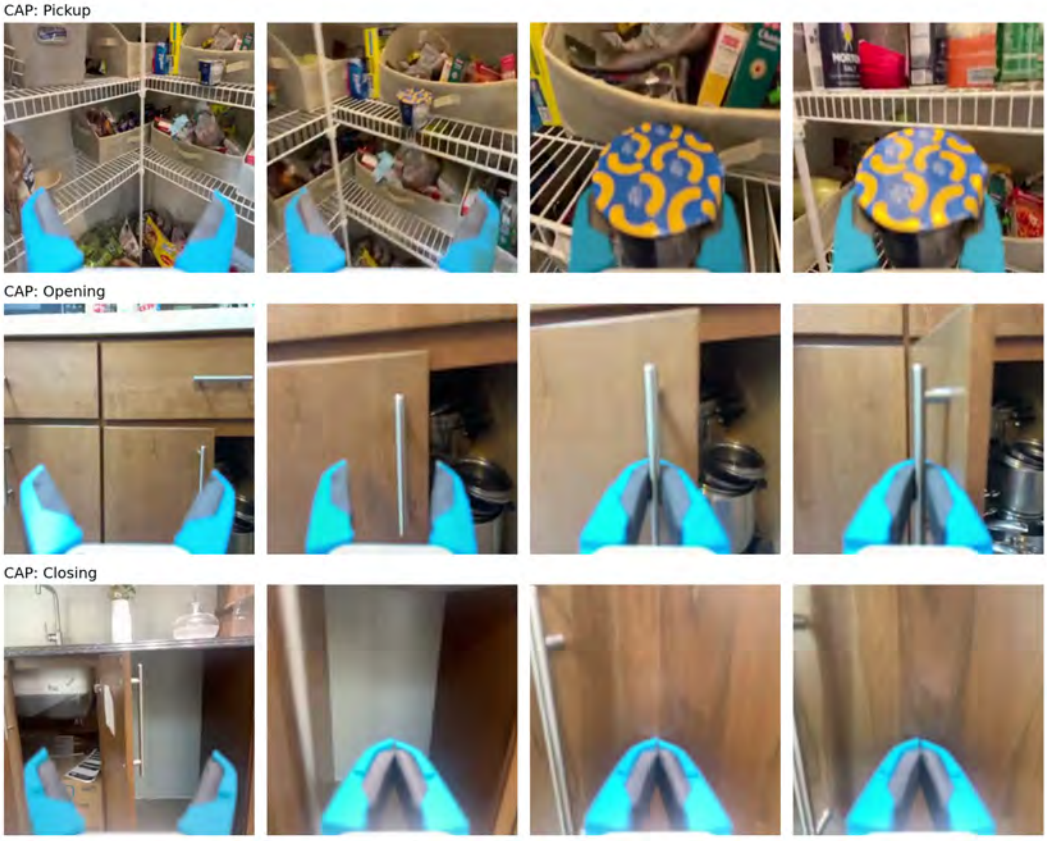

Figure 13: CAP dataset pick, open, and close trajectory samples.