AutoSpeed: Annotation-Free Stage-Adaptive Motion Speed Learning for Robot Manipulation

Qingda Hu[1*], Ziheng Qiu[1*], Jieru Zhao[2], Zhongxue Gan[1], and Wenchao Ding[1†]

[1] College of Intelligent Robotics and Advanced Manufacturing, Fudan University, Shanghai, China

[2] School of Computing, Shanghai Jiao Tong University, Shanghai, China

[*]Equal contribution. [†]Corresponding author: dingwenchao@fudan.edu.cn

Abstract. Different stages of manipulation tasks exhibit varying levels of difficulty, suggesting stage-dependent motion speeds and temporal prediction horizons. However, existing IL-based visuomotor policies typically imitate the execution speed of expert demonstrations and operate with a fixed temporal prediction horizon, limiting flexibility and overall task throughput. In this paper, we introduce **AutoSpeed**, a model-agnostic learning framework that enables existing visuomotor policies to predict trajectories with stage-adaptive motion speeds, without requiring speed or stage annotations. We treat future trajectories at different speeds as candidate optimization targets, evaluate each candidate using a composite cost that trades off prediction error against prediction horizon, and optimize the policy toward the minimum-cost candidate. With a fixed-length action sequence, speed modulation adjusts the effective temporal prediction horizon: simple stages are executed faster with a longer prediction horizon, whereas complex stages are executed more slowly with a shorter prediction horizon. Specifically, we implement speed modulation in the frequency domain via the discrete cosine transform (DCT), which enables smooth, non-integer speed scaling and thus preserves motion continuity. Extensive evaluations show that AutoSpeed substantially reduces task execution time while also improving success rates. Under the AutoSpeed framework, the inferred motion speeds exhibit a strong correspondence with task stages.

Keywords: Imitation Learning · Adaptive Motion Speed · Visuomotor Policy Learning

1 Introduction

Imitation learning (IL) is widely adopted for visuomotor policy learning. In recent years, it has catalyzed a diverse range of visuomotor policies [8, 13, 21, 42] for robotic manipulation, including prominent Vision-Language-Action (VLA) models [3,11,24,39]. Existing IL-based policies typically mimic the motion speed

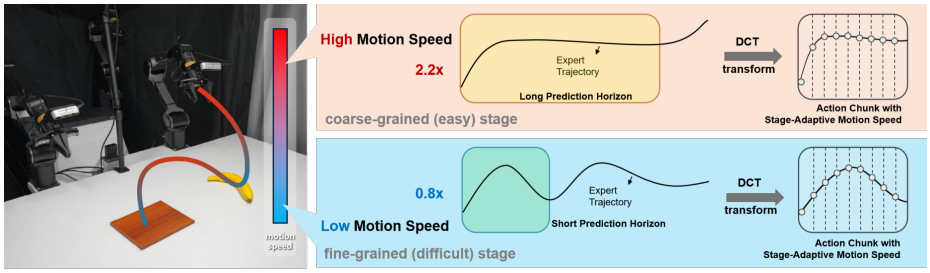

Fig. 1: Stage-aware motion speed adaptation. Motion speeds in expert demonstrations are often suboptimal. AutoSpeed aims to train policies to predict future trajectories with stage-aware motion speed without requiring speed or stage annotations.

exhibited in expert demonstrations. [6] Regardless of the stage of the task, the policy predicts actions over a fixed future horizon, i.e., it uses fixed-length action chunking [8, 40]. However, the motion speeds in expert demonstrations are often suboptimal, thereby limiting the efficiency and performance of the learned policy. In practical deployment scenarios, task completion efficiency is often critical, especially in industrial settings.

Different stages of manipulation tasks exhibit varying levels of difficulty [7, 33], suggesting that both the motion speed and the temporal prediction horizon should be stage-dependent. Empirically, we find a consistent positive relationship between motion speed and effective prediction horizon, aligning with prior findings in human motor and cognitive control [9, 28]. As shown in Fig. 1, in easy stages (e.g., free-space reaching, or gross repositioning), long-horizon action chunks can be predicted reliably from the current observation [25, 35], allowing faster execution. In contrast, fine-grained stages (e.g., insertion, or sustained interaction) demand higher precision and tighter perception–action coupling, and are therefore better served by shorter effective prediction horizons and slower execution. This is consistent with findings in human motor and cognitive control: when task demands are low, humans can act faster with relatively little monitoring, whereas precision-critical behaviors require closer feedback monitoring and often slower movements to preserve accuracy [2, 15, 29].

Building on these motivations, we introduce AutoSpeed, a model-agnostic learning framework that enables existing visuomotor policies to predict trajectories at stage-adaptive motion speeds. In this work, we hypothesize and validate that the stage-dependent speed ratio can be implicitly inferred during end-to-end policy training. This obviates the need to train an additional proxy policy [12] or to rely on VLM-based or manual annotations [10, 18, 19]. Moreover, AutoSpeed jointly optimizes the quality of implicit speed ratio inference and trajectory prediction in an end-to-end training pipeline.

Concretely, as shown in Fig. 1, given a set of speed ratios (e.g., sampled from 0.8 to 2.2 in increments of 0.2), we obtain action chunks at different speeds by applying discrete cosine transform (DCT) [17] to the original trajectories in the frequency domain. This technique enables non-integer speed modulation while

preserving more high-frequency action details for fine-grained manipulation [41]. We use fixed-length action chunks to keep the model's output dimensionality consistent; acceleration compresses trajectories in time and increases the effective horizon, while deceleration expands them and reduces it. We then treat future trajectories at different speeds as a set of candidate supervision targets. Each candidate is evaluated with a composite cost that trades off prediction error against prediction horizon, and we optimize the policy toward the minimum-cost candidate (section 3.2). A lightweight Ratio Head is trained jointly to predict the speed ratio from latent observation features. Additionally, similar to the temporal ensemble [40], we introduce a nonlinear temporal ensemble (section 3.4) that outputs actions by combining the speed prediction of the ratio head with multiple overlapping fragment predictions, allowing for smooth deployment at inference time.

Through extensive experiments across diverse simulation benchmarks and real-world tasks, we show that visuomotor policies trained with AutoSpeed consistently reduce task execution time while improving success rates. AutoSpeed supports both single-task and multi-task learning and works across policy classes, spanning non-generative (e.g., MLP-based) and generative (e.g., diffusion- and flow-based) action prediction models. We further demonstrate additional capabilities of AutoSpeed, including controllable motion style and improved policy performance on datasets with substantial demonstration-speed variability. We summarize the contributions of this paper as follows:

1. We introduce AutoSpeed, which to our knowledge is the first annotation-free, stage-aware framework to implicitly infer motion speed ratios within end-to-end policy learning.
2. With DCT-based frequency-domain scaling, AutoSpeed enables non-integer acceleration and deceleration while preserving high-frequency action details.
3. Policies trained with AutoSpeed substantially shorten task completion time while improving success rates. Moreover, the inferred speed ratios are closely aligned with task stages.

2 Related Work

2.1 Motion Speed Modulation in Robot Manipulation

We categorize the literature on motion speed modulation in robot manipulation by when the motion speed ratio is inferred: (1) pre-training inference, where it is inferred before training and used to annotate the dataset; (2) test-time inference, where it is inferred during model deployment; and (3) training-time emergence from unlabeled data, where it is implicitly inferred during end-to-end training.

The following methods fall into the first category: DemoSpeedUp [12] derives safe-to-accelerate regions by estimating action entropy with a proxy policy, and retrains a faster policy on the annotated dataset. Similarly, ESPADA [18] uses VLM/LLM-based stage annotations to identify the safe-to-accelerate regions. Moreover, SpeedAug [27] constructs tempo-augmented trajectories to learn a

tempo-enriched imitation prior, then applies RL-based fine-tuning to push toward faster policies. And the following methods fall into the second category: SAIL [1] integrates complexity-aware speed modulation with high-gain tracking and scheduling to safely increase execution speed, while SRIL [36] reduces computing latency by learning when to skip inference. VFIL [26] treats speed as an explicit command and conditions the policy on sampling frequency, enabling variable-speed execution by changing the control frequency at deployment without retraining.

However, the third paradigm remains underexplored despite its practical importance. Annotation-based approaches face two key limitations. First, introducing external tools or training additional models to produce annotations is costly and often requires task-specific heuristics or rules. Second, annotation quality and cross-trajectory consistency directly affect the final policy performance, yet these annotations are not optimized end-to-end with the policy. Moreover, prior work largely focuses on when to accelerate; in AutoSpeed, we unify acceleration and deceleration within a single speed-modulation perspective.

2.2 Flexible Reconfiguration of Action Chunking

Action chunking, which models the joint distribution of future actions conditioned on past states, is a standard practice in robotic manipulation [8, 40]. While action chunking strengthens temporal consistency and mitigates error accumulation, long chunks reduce access to the most recent observations during execution and thus limit reactivity [25]. A growing body of work shows that the prediction horizon in action chunking can substantially impact policy performance, suggesting the existence of an optimal horizon [4, 25, 34]. Embodied manipulation is inherently stage-structured: predictability and feedback requirements vary substantially across stages, so a single global optimal horizon is at best an average-case compromise. A more appropriate view is local optimality, where the effective horizon should be adapted to the current context during policy prediction or execution.

From the training-time perspective, MoH [16] formulates action chunking as a mixture over multiple horizons, processes them in parallel with a shared action transformer, and fuses the outputs via a gating module. From the inference-time perspective, AutoHorizon [34] adjusts the execution horizon online based on the model's attention weights. GBC [32] improves diffusion-based behavior cloning via self-guidance adaptive chunking mechanism that selectively refreshes action chunks when higher reactivity is needed.

In contrast to these horizon-adaption designs, AutoSpeed does not explicitly enumerate or select action chunks with different horizons. Instead, the policy is directly optimized toward the optimal action target, avoiding the need to model suboptimal horizon modes or incur the inference-time cost of grouped prediction. Moreover, the speed ratio emerges implicitly during policy learning, making the mechanism model-agnostic and easy to apply across policy architectures.

The shared motivation between motion speed modulation and action chunking reconfiguration is to move beyond a single globally tuned trade-off toward

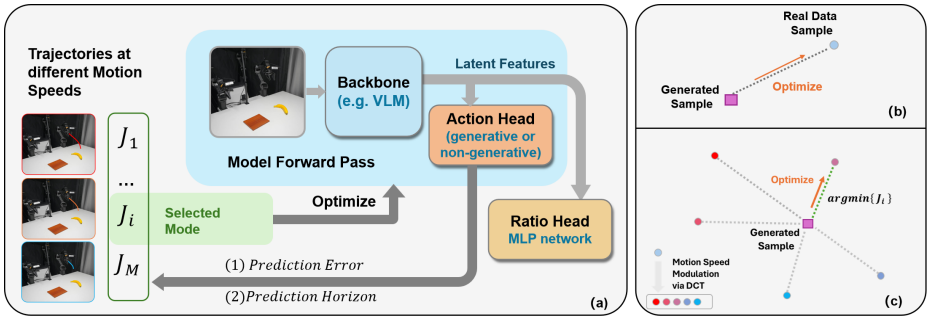

Fig. 2: Overview of AutoSpeed. Training with AutoSpeed is formulated as a cost-aware multi-target selective optimization problem, as illustrated in (a), where trajectories at different motion speeds form a set of candidate supervision targets. AutoSpeed performs mode selection by minimizing a composite cost J that trades off prediction error against prediction horizon, and optimizes the policy toward the minimum-cost target. (b) and (c) contrast generative-model training with and without AutoSpeed. With AutoSpeed (c), each sample is guided toward the target that attains the lowest cost.

stage-aware, locally optimal adaptation across task stages. Dynamic adjustment of the prediction horizon covered by action chunks is also one of our motivations, and we find it is strongly positively correlated with motion-speed modulation across task stages. Accordingly, we keep the output chunk size fixed to preserve architectural compatibility, and use motion-speed modulation to induce a coupled change in the effective inference horizon, enabling stage-adaptive temporal reasoning without altering the model output dimensionality.

3 Method

3.1 Problem Setup

Imitation learning for robotic manipulation aims to train a policy by minimizing the prediction error of demonstrated trajectories conditioned on observations. An expert dataset is given as $\mathcal{D} = \{(l^{(i)}, \tau^{(i)})\}_{i=1}^{N},$, where each trajectory τ is denoted by $\tau = \{(o_t, a_t, s_t)\}_{t=1}^{T}$ and the task instruction l specifies the task. At each time step t, o_t denotes the image observation, a_t denotes the motion control signal, and s_t denotes the robot state. We denote the chunk size by H, and the observation horizon by K. For brevity, we omit conditioning inputs other than $\mathbf{o}_{t-K+1:t}$. A standard visuomotor policy π_θ predicts a length-H action chunk conditioned on the recent observation context:

$$\pi_\theta(\mathbf{a}_{t:t+H-1} \mid \mathbf{o}_{t-K+1:t}) \,. \tag{1}$$

To enable stage-adaptive motion speed learning, the key challenge is to identify which stages in demonstration trajectories can be safely accelerated. We define the motion speed ratio as r_t at time step t. Larger r_t corresponds to easier stages, allowing faster execution and a longer temporal prediction horizon;

smaller r_t corresponds to precision-critical stages, demanding slower execution, a shorter horizon and higher-frequency feedback.

In our framework we use a fixed number of chunks to keep the model's output tensor shape consistent. Therefore, the temporal prediction horizon h_t is positively correlated with the motion speed ratio r_t. then:

$$\mathrm{h}_t \;=\; H \cdot r_t \tag{2}$$

Accordingly, a visuomotor policy trained with AutoSpeed is formulated as:

$$\pi_\theta(\mathbf{a'}_{t:t+h_t-1}, \mid \mathbf{o}_{t-K+1:t})\,. \tag{3}$$

where $\mathbf{a'}_{t:t+h_t-1}$ has a motion speed r_t matched to the current task stage. The transformation from $\mathbf{a}_{t:t+H-1}$ to $\mathbf{a'}_{t:t+h_t-1}$ is described in Sec. 3.3. Besides, Sec. 3.2 details the AutoSpeed optimization procedure. Sec. 3.4 presents training recipes for generative policy classes and introduces a compatible inference-time strategy that aggregates overlapping predictions to improve smoothness. In the following sections, we denote an action chunk by A.

3.2 Multi-Target Selective Optimization

Recent studies have analyzed and empirically validated quantifiable stage signatures in robotic manipulation: fine-grained behaviors are often reflected in the high-frequency components of action trajectories [41], while the policy's predicted action entropy can serve as a proxy for identifying difficult, failure-prone stages [12, 14, 31]. We hypothesize that the stage-dependent signal r_t can be implicitly inferred during end-to-end training, and in this work we validate that it is feasible and model-agnostic. We formulate training as cost-aware multi-target optimization.

Optimization Target Set At each time step t, we construct a set of candidate future trajectories at different speeds, denoted by $\{A^{(m)}\}_{m=1}^{M}$. AutoSpeed supports a user-defined discrete set of motion-speed ratios (e.g., sampled from 0.8 to 2.2 in increments of 0.2), and M denotes the number of speed candidates. Compared to manually downsampling demonstrations at only integer-rate factors [12, 18], this design provides greater flexibility and enables smoother speed transitions.

Cost-based Selective Optimization As shown in Fig. 2, the multi-target setting yields a set of losses, each corresponding to an optimization direction. We evaluate these candidates using a composite cost J that trades off prediction error against temporal prediction horizon:

$$J\left(A^{(m)}\right) = \frac{\mathcal{E}_t^{(m)}}{w + \log(h_t)}, \tag{4}$$

$\mathcal{E}_t^{(m)}$ denotes the model's mean-squared prediction error (MSE) for candidate m at time t. For non-generative models, $\mathcal{E}_t^{(m)}$ reduces to the MSE between the predicted action chunk and $A^{(m)}$. For generative models, $\mathcal{E}_t^{(m)}$ can be instantiated as an MSE either on the denoising target (i.e., the injected noise) or directly on the ground-truth actions [22], with the former being more commonly used [8,24,38]. The denominator serves as an adaptive normalization term. Candidates with shorter prediction horizons receive a larger penalty, encouraging the model to favor well-predicted futures while selecting faster speeds whenever possible. Intuitively, in easier stages, future actions remain predictable over longer horizons, permitting higher speed ratios; in precision-critical stages, long-horizon prediction becomes less predictable, yielding lower speed ratios. Additionally, w is a constant term that can be tuned to control motion style. A smaller w biases the policy toward a more aggressive motion style.

We then select the minimum-cost candidate:

$$m^\star = \arg\min_{m\in\{1,\dots,M\}} J\Big(\mathbf{A}^{(m)}\Big) \tag{5}$$

AutoSpeed trains the policy by minimizing the loss on the selected target,

$$\min_\theta \ \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_t \mathcal{L}_\theta\Big(\mathbf{o}_{t-K+1:t},\ \mathbf{A}_t^{m^\star}\Big)\right]. \tag{6}$$

Criterion for different models. AutoSpeed is model-agnostic. For policies with non-generative action heads, the model output is the action prediction; therefore, we set $\mathcal{E}_t^{(m)}$ to the mean-squared error (MSE) of the predicted actions.

For diffusion-based action heads, the action chunk $A^{(m)}$ serves as the generation target and is progressively corrupted by Gaussian noise $\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ during training to obtain a noised sample $\mathbf{x}_\kappa^{(m)}$ at diffusion step κ; the denoiser $\epsilon_\theta(\cdot)$, conditioned on the observation context $\mathbf{o}_{t-K+1:t}$, learns to predict the injected noise. Therefore, we set $\mathcal{E}_t^{(m)}$ to the mean-squared error (MSE) of the predicted noise:

$$\mathcal{E}_t^{(m)} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\,\kappa}\left[\left\|\boldsymbol{\epsilon} - \epsilon_\theta\Big(\mathbf{o}_{t-K+1:t},\ \mathbf{x}_\kappa^{(m)},\ \kappa\Big)\right\|_2^2\right], \tag{7}$$

For flow-matching-based action heads, the model learns a conditional velocity field $v_\theta(\cdot)$ that transports a noise sample $\boldsymbol{\epsilon}$ to the target action chunk $A^{(m)}$ along an interpolation $\mathbf{x}_\tau^{(m)} = (1-\tau)\boldsymbol{\epsilon} + \tau A^{(m)}$, where $\tau\in[0,1]$ is the interpolation time. Accordingly, we set

$$\mathcal{E}_t^{(m)} = \mathbb{E}_{\tau\sim\mathcal{U}(0,1),\,\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\left\|v_\theta\Big(\mathbf{o}_{t-K+1:t},\ \mathbf{x}_\tau^{(m)},\ \tau\Big) - \Big(A^{(m)} - \boldsymbol{\epsilon}\Big)\right\|_2^2\right]. \tag{8}$$

Ratio Head. We introduce a lightweight Ratio Head trained to predict the speed ratio r_t from latent observation features. It provides stage-adaptive motion speeds in the third stage of generative-model training 3.4, and its predictions are used by the Nonlinear Temporal Aggregation (NTA) module 3.4 during inference.

3.3 Motion Speed Transform

We apply the discrete cosine transform (DCT) [17] to map time-domain action signals into the frequency domain, perform temporal scaling there to realize stretching/compression along the time axis, and then transform the actions back to the time domain.

In details, we first compute DCT-II coefficients of the original action chunk $A \in \mathbb{R}^{H_0 \times D}$ along the temporal axis, where H_0 denotes the maximum number of action samples required:

$$\mathbf{C} = \mathrm{DCT}(A) \in \mathbb{R}^{H_0 \times D} \tag{9}$$

We then retime by evaluating the inverse-DCT basis at the speed ratio r and reconstruct the retimed chunk via a basis–coefficient product:

$$\tilde{A} = \mathbf{B}(r)\mathbf{C}, \qquad \mathbf{B}(r)_{i,k} = \sqrt{\frac{2}{H_0}}\,\alpha_k \cos\Big(\frac{\pi k(t_i + 0.5)}{H_0}\Big), \tag{10}$$

where $t_i = i \cdot r$ for $i = 0, \ldots, H-1$, $k = 0, \ldots, H_0 - 1$, $\alpha_0 = 1/\sqrt{2}$, and $\alpha_k = 1$ for $k \geq 1$. During training, the candidates $\{\tilde{A}^{(m)}\}_{m=1}^{M}$ are generated by applying the operator with the speed ratio set $\{r_m\}_{m=1}^{M}$.

This enables both acceleration and deceleration with non-integer scaling factors, while the frequency-domain processing effectively preserves high-frequency details that are crucial for fine-grained manipulation [41].

3.4 Other Techniques

Training Recipe for Generative Models For non-generative models, applying AutoSpeed introduces no additional model computation overhead. For generative models, AutoSpeed incurs a small additional computational cost: the model's action head needs to denoise each noise-perturbed trajectory candidate. However, this overhead is limited for two reasons. First, the parameters of the action head typically constitute only a small fraction of the overall model [3,13,39]. Second, the denoising of different trajectory candidates can be performed in parallel [20,30], which prevents the training process from being significantly slowed down.

To further improve training stability for generative models and reduce the additional computational cost, we propose a three-stage training strategy: Inspired by [5], which shows that diffusion models tend to preferentially learn smooth, low-complexity, and generalizable structure in the early stages of training, we optimize the model in the early training stage by minimizing the average loss over all candidates. The second stage corresponds to the multi-objective optimization process (Sec. 3.2). In the third stage, the Ratio Head is frozen, and the model is optimized directly using the motion speed predicted by the Ratio Head, allowing the remaining steps in the generation process to rapidly converge to the selected mode.

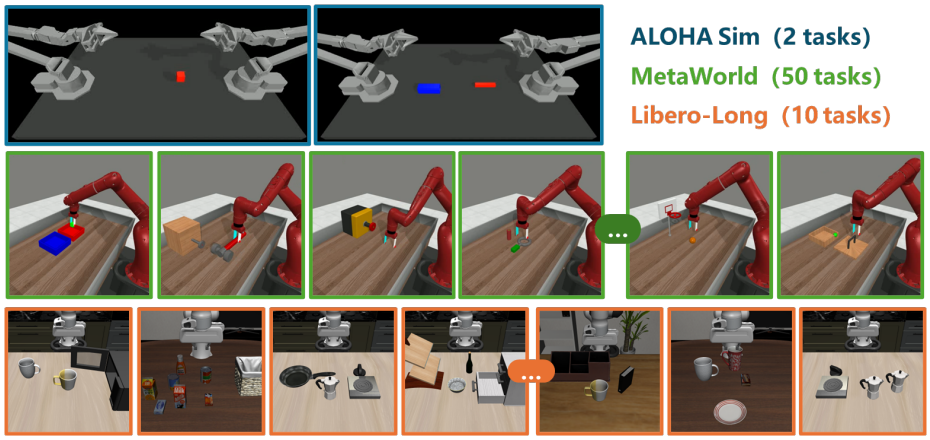

Fig. 3: Simulation Tasks. We select a total of 62 tasks from ALOHA Simulation [40], MetaWorld [37], and LIBERO-Long [23], and use 50 demonstration trajectories for each task.

Nonlinear Temporal Aggregation Conventional temporal aggregation assumes uniform time intervals consecutive actions across all chunks, rendering it incompatible with the variable-speed predictions generated by AutoSpeed. To address this limitation, we introduce Nonlinear Temporal Aggregation (NTA), a novel mechanism that enables the ensembling of overlapping prediction across action chunks operating at different temporal scales.

NTA computes a weighted average of actions from multiple chunks within a predefined time window. Specifically, at a given control step s (denoted temporally as $t = 0$), all predicted actions from previously generated chunks that fall within the time window $[-i, +i]$ are collected. Then, we compute a weighted average of the candidate actions using an exponential-decay weighting scheme and slight temporal misalignment is also corrected via the same frequency-domain transformation.

4 Evaluation

Our evaluations aim to investigate and answer the following questions:

1. Can AutoSpeed effectively reduce the time required to complete tasks while maintaining task success rates?
2. Can the inferred speed ratios produced by AutoSpeed effectively indicate different task stages?
3. Is AutoSpeed effective in both single-task and multi-task learning settings, and does it work for both non-generative and generative models?

4.1 Experimental Setup

We evaluate AutoSpeed across a range of standard manipulation benchmarks and real-world tasks.

Simulation Tasks: We experiment with two bimanual tasks from the ALOHA benchmark [40], 10 tasks from the LIBERO-10 suite [23], 50 tasks from the Meta-World benchmark [37], shown in Fig. 3. In ALOHA simulation, we evaluate the Transfer Cube and Insertion tasks, utilizing 50 expert demonstrations per task. For both the LIBERO-10 suite and the Meta-World benchmark, the policies are trained using 50 expert demonstrations per task under a multi-task learning paradigm.

Real-world Tasks: We conduct real robot experiments on an Agilex Piper bimanual robot platform in a tabletop manipulation environment. The policies are trained on RGB images and robot proprioceptive state. The images are captured via two wrist cameras mounted on the top of the grippers and one fixed camera installed on top of the table. The action space comprises the joint states and the gripper state. Using a teleoperation system, we collect a dataset for four distinct tabletop tasks, with each task comprising approximately 50 expert demonstrations. Notably, the expert action trajectories are temporally oversampled by a factor of 2, enabling fine-grained actions when deploying deceleration.

Models We compare policies optimized with our AutoSpeed framework against their vanilla counterparts trained with original objectives.

ACT : Action Chunking Transformer(ACT) [40] is a visuomotor policy composed of a transformer encoder and decoder. We use it as a representative non-generative baseline model.

BAKU : BAKU [13] is a transformer-based architecture designed for multi-task learning. This architecture is highly representative: most existing visuomotor policies, including VLA models, are largely built upon an Observation Encoder–Backbone–Action Head design. It employs a transformer encoder to fuse information from different modalities and trains a decoupled action head to generate action chunks. This modular design enables the action head to be readily substituted with several generative variants.

Metrics In addition to the standard metric of task success rate, we report the average episode length of successful rollouts to assess overall execution efficiency. For the Aloha benchmark, we perform 50 evaluation rollouts per task. For both the LIBERO-10 suite and the Meta-World benchmark, we adopt a consistent evaluation protocol, executing 10 trials per task and reporting the aggregated success rates and execution length. For the real-world evaluation, we report the task success rate and average completion time (in seconds) across 20 trials per task.

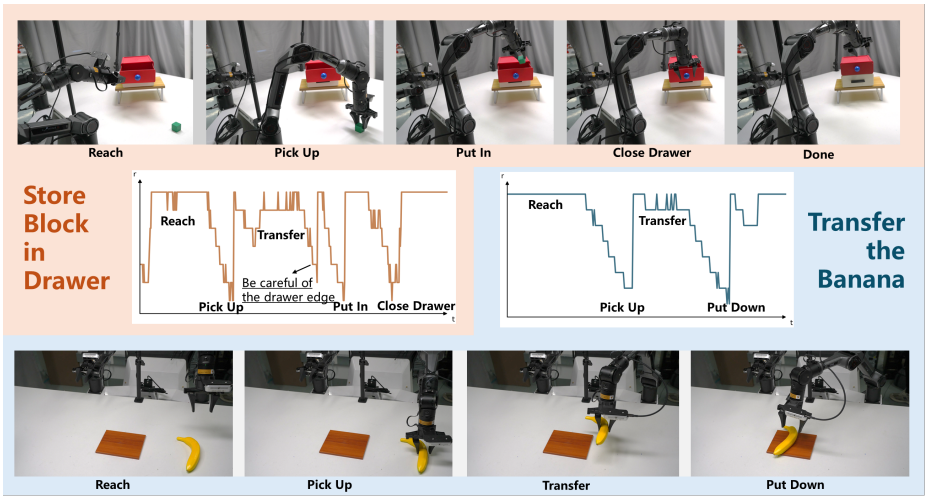

Fig. 4: The speed ratio curves of two real-world tasks correspond to the task stages. Under the AutoSpeed framework, the inferred speed ratios closely align with task stages.

4.2 Main Results on Single-Task Learning Simulation

Table 1 summarizes the quantitative results evaluated on the ALOHA benchmark.

Compared to the vanilla ACT baseline, the policy trained with AutoSpeed (denoted as *AutoSpeed*) achieves a substantial reduction in task execution time without compromising overall performance. Specifically, on the Transfer Cube task, AutoSpeed reduces the average episode length from 272 to 160 steps, yielding a maximum speedup of 1.7x. While this accelerated execution slightly lowers the success rate (from 72% to 64%) under standard control due to the heightened sensitivity of high-speed interactions, coupling AutoSpeed with a high-gain controller effectively mitigates this issue. This combination not only preserves the execution efficiency but also surpasses the original baseline with a 6% improvement. Furthermore, on the contact-rich Insertion task, AutoSpeed simultaneously improves the success rate (from 22% to 24%) and reduces execution length (from 353 to 296 steps with high-gain), demonstrating its robust execution efficiency in single-task learning scenarios.

4.3 Main Results on Multi-Task Learning Simulation

The quantitative results on the LIBERO-10 and Meta-World benchmark are presented in Table 2. AutoSpeed consistently outperforms the strongest baselines across both multi-task benchmarks. On Meta-World, AutoSpeed yields substantial improvements in success rates regardless of the underlying action head: BAKU-DiT exhibits a 19.0% increases, while BAKU-Flow improves by 7.8% to

Table 1: Quantitative results across multiple tasks on the ALOHA benchmark. † denotes evaluated with a high-gain controller.

Method	Transfer Cube		Insertion	
	SR (↑)	**Len** (↓)	**SR** (↑)	**Len** (↓)
ACT	72%	272	22%	353
ACT(*AutoSpeed*)	64%	**160**	**24%**	300
ACT(*AutoSpeed*†)	**78%**	165	**24%**	**296**

achieve the highest overall success rate (65%). Crucially, these gains are accompanied by marked reductions in average episode lengths. On the LIBERO-10 suite, AutoSpeed-trained policy compresses the execution time with a maximum acceleration of approximately 40% to its vanilla counterpart. Compared to baselines that naively execute actions at a fixed, dataset-dictated speed, AutoSpeed's stage-adaptive optimization empowers policies to complete tasks substantially faster, while simultaneously improving success rates by dynamically adjusting to the complexity of the current task phase.

Crucially, our experiments demonstrate that reducing task execution time and improving success rates are not mutually exclusive. By predicting trajectories at stage-adaptive motion speeds, AutoSpeed leverages long-horizon action chunks for faster execution during easy stages, where future actions can be predicted reliably. Conversely, during fine-grained stages that demand tighter perception-action coupling, the framework naturally adopts slower execution and shorter effective horizons to preserve control accuracy. Because this stage-aware modulation is implicitly inferred during end-to-end training via our composite cost, AutoSpeed generalizes effectively across multi-task scenarios, delivering superior execution efficiency while preserving task success compared to baselines.

Beyond qualitative observations, we further analyze the consistency between the predicted speed ratios and action entropy from DemoSpeedUp [12], which reflects divergence across multiple sampled action predictions, and find that unlike action entropy, which relies on grouped predictions from a proxy policy and struggles to distinguish uncertainty from multi-modality, AutoSpeed more accurately identifies safely acceleratable stages, such as robot start-up and return-to-home fast motions.

4.4 Main Results on Real-World Tasks

As shown in Table 3, AutoSpeed consistently outperforms the baselines across all evaluated real-world tasks with various length and difficulty levels. Most notably, when integrated with our proposed Nonlinear Temporal Aggregation (NTA), the AutoSpeed-trained policy significantly reduces execution time, delivering an average speedup of approximately 1.78× across the diverse task suite.

Crucially, this substantial acceleration does not compromise manipulation precision but yields consistent absolute improvements in task success rates. As

Table 2: Quantitative results on multi-task learning benchmarks. Success rate (SR, ↑) and episode length (Len, ↓).

Method	LIBERO-10		Meta-World	
	SR (↑)	**Len** (↓)	**SR** (↑)	**Len** (↓)
BAKU-DiT	50%	261	43%	87
BAKU-Flow	47%	287	57%	82
BAKU-DiT (*AutoSpeed*)	49%	259	62%	**70**
BAKU-Flow (*AutoSpeed*)	**52%**	**171**	**65%**	71

Table 3: Real-world evaluation. Success rate (SR, ↑) and execution time in seconds (Time, ↓).

Method	Transfer the Banana		Place the Toy	
	SR (↑)	**Time** (↓)	**SR** (↑)	**Time** (↓)
BAKU-Flow+TA	77%	20.80s	33%	18.13s
BAKU-Flow (*AutoSpeed*)+NTA	**83%**	**11.24s**	**43%**	**10.13s**
Method	**Stacking Two Cubes**		**Store Block in Drawer**	
	SR (↑)	**Time** (↓)	**SR** (↑)	**Time** (↓)
BAKU-Flow+TA	60%	50.11s	77%	38.67s
BAKU-Flow (*AutoSpeed*)+NTA	**63%**	**28.52s**	**80%**	**23.01s**

illustrated in Fig. 4, in the highly challenging *Place the Toy* task, AutoSpeed increases the success rate by a notable 10%. These physical deployments further validate AutoSpeed's capacity to autonomously extract stage-dependent motion speed patterns from unannotated demonstrations. By leveraging the proposed multi-target optimization, the policy translates these learned patterns into superior task efficiency and robust real-world performance.

Notably, AutoSpeed trained on non-oversampled data still outperforms the baselines, experiencing only a marginal performance drop compared to the oversampled version. Since action oversampling is practically feasible, it simply serves as an optional enhancement to further improve fine-grained action fidelity enabled by DCT-based retiming.

4.5 Ablation Studies

Speed Range We evaluate how different predefined speed ranges impact the model's ability to learn phase-dependent motion patterns and its overall task performance. Specifically, we train the ACT policy equipped with AutoSpeed on the ALOHA Transfer Cube task under three distinct speed range settings: AutoSpeed [1x, 2x], AutoSpeed [1x, 4x], and AutoSpeed [2x, 4x], and a vanilla

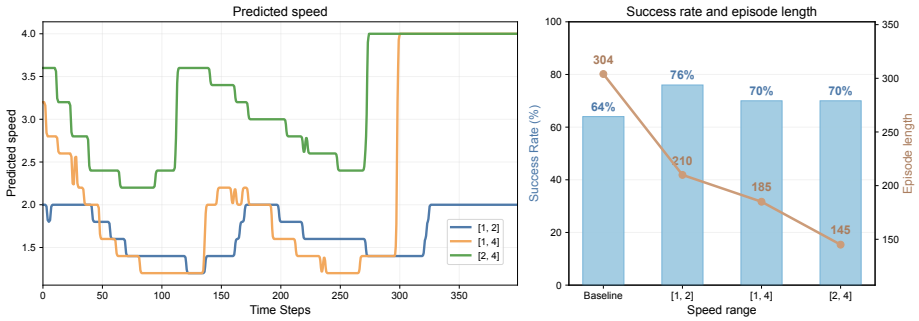

Fig. 5: Ablation on Speed Range Bounds. Left: Predicted motion speed trajectories over time steps for AutoSpeed variants on the ALOHA Transfer Cube task. Despite different predefined bounds, all variants exhibit a consistent phase-aware pattern, autonomously decelerating during complex interaction stages. **Right:** Comparison of task success rates and average episode lengths with all variants outperforming the vanilla baseline.

ACT model is included as the baseline. Note that while this ablation uses a reduced image resolution compared to the main experiments, the same resolution and all other hyperparameters are kept identical across all variants to ensure a fair comparison.

As illustrated in Fig. 5(left), despite the varying upper and lower bounds, the learned speed trajectories across all three variants exhibit a remarkably consistent phase-aware pattern. The policies autonomously decelerate (forming a valley in the speed curve) during complex, interaction-critical task phases, and accelerate during simpler, unconstrained stages.

Furthermore, Fig. 5(right) demonstrates that all three AutoSpeed variants consistently outperform the vanilla baseline in both success rate and execution efficiency. Notably, the results reveal a clear tradeoff dictated by the speed bounds: the more conservative [1x, 2x] setting achieves the highest success rate (76% vs. the baseline's 64%), while the aggressive [2x, 4x] setting maximally minimizes the episode length (dropping from 304 to 145 steps) while still maintaining a highly competitive success rate of 70%.

Sensitivity to w The speed range and w are both practical design choices for specific task requirements. Different length penalty coefficients inherently shift the model's preference for speed selection. As the penalty decreases, the relative cost of predicting shorter action chunk increases. Consequently, the model is penalized more heavily for selecting slower speeds and tends to predict action chunk with longer horizons. Fig. 6 shows as policy trained with a smaller length penalty favors generating action chunk with higher speed. Overall, different choices share the phase-aware pattern: the policy accelerates in predictable stages and decelerates in critical interaction stages.

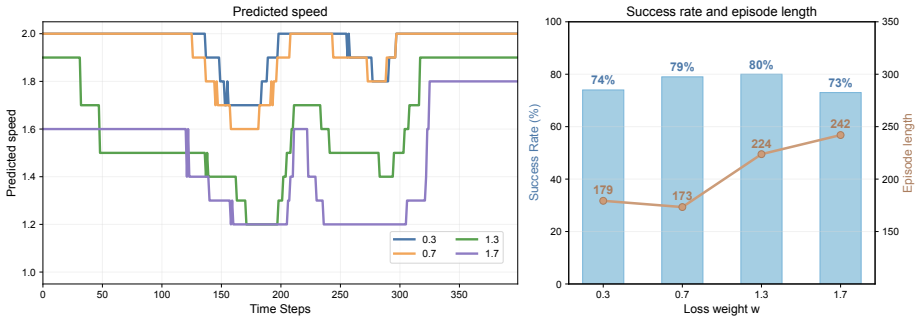

Fig. 6: Ablation on the length penalty coefficient w. Left: Predicted motion speed trajectories over time steps for AutoSpeed variants on the ALOHA Transfer Cube task. **Right:** Comparison of task success rates and average episode lengths.

5 Conclusion

AutoSpeed is a simple yet effective training paradigm for learning stage-adaptive motion speed and temporal prediction horizon directly from standard expert demonstrations. Rather than relying on external annotations or post-hoc trajectory retiming, AutoSpeed uncovers an endogenous signal r_t during end-to-end policy learning by selecting, among re-timed candidate futures, the target that best balances prediction error and temporal prediction horizon. This unifies speed control and temporal reasoning under a single objective and is compatible with both non-generative and generative visuomotor policies. Across a broad suite of simulation and real-world manipulation tasks, AutoSpeed improves efficiency while maintaining success rates. We hope AutoSpeed offers a practical and broadly applicable route toward faster, safer, and more reliable visuomotor policy deployment.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62403142, and in part by the Science and Technology Commission of Shanghai Municipality under Grant 24511103100.

References

1. Arachchige, N.R., Chen, Z., Jung, W., Shin, W.C., Bansal, R., Barroso, P., He, Y.H., Lin, Y.C., Joffe, B., Kousik, S., Xu, D.: Sail: Faster-than-demonstration execution of imitation learning policies. In: Proceedings of The 9th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 305, pp. 721–749 (27–30 Sep 2025)
2. Badre, D.: Cognitive control. Annual Review of Psychology **76**(1), 167–195 (2025)

3. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: $\pi 0$: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164 (2024)
4. Black, K., Galliker, M., Levine, S.: Real-time execution of action chunking flow policies. In: Advances in Neural Information Processing Systems. vol. 38, pp. 33383–33407 (2025)
5. Bonnaire, T., Urfin, R., Biroli, G., Mézard, M.: Why diffusion models don't memorize: The role of implicit dynamical regularization in training. arXiv preprint arXiv:2505.17638 (2025)
6. Brown, D.S., Goo, W., Niekum, S.: Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In: Conference on robot learning. pp. 330–359. PMLR (2020)
7. Chen, Q., Yu, J., Schwager, M., Abbeel, P., Shentu, Y., Wu, P.: Sarm: Stage-aware reward modeling for long horizon robot manipulation. arXiv preprint arXiv:2509.25358 (2025)
8. Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. The International Journal of Robotics Research **44**(10-11), 1684–1704 (2025)
9. Elliott, D., Helsen, W.F., Chua, R.: A century later: Woodworth's (1899) two-component model of goal-directed aiming. Psychological bulletin **127**(3), 342 (2001)
10. Fan, Y., Bai, S., Tong, X., Ding, P., Zhu, Y., Lu, H., Dai, F., Zhao, W., Liu, Y., Huang, S., et al.: Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. In: Conference on Robot Learning. pp. 2018–2037. PMLR (2025)
11. Ghosh, D., Walke, H.R., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y.L., Chen, L.Y., Vuong, Q., Xiao, T., Sanketi, P.R., Sadigh, D., Finn, C., Levine, S.: Octo: An open-source generalist robot policy. In: Proceedings of Robotics: Science and Systems. Delft, Netherlands (Jul 2024). https://doi.org/10.15607/RSS.2024.XX.090
12. Guo, L., Xue, Z., Xu, Z., Xu, H.: Demospeedup: Accelerating visuomotor policies via entropy-guided demonstration acceleration. In: Proceedings of The 9th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 305, pp. 599–609 (27–30 Sep 2025)
13. Haldar, S., Peng, Z., Pinto, L.: Baku: An efficient transformer for multi-task policy learning. Advances in Neural Information Processing Systems **37**, 141208–141239 (2024)
14. Hejna, J., Mirchandani, S., Balakrishna, A., Xie, A., Wahid, A., Tompson, J., Sanketi, P., Shah, D., Devin, C., Sadigh, D.: Robot data curation with mutual information estimators. arXiv preprint arXiv:2502.08623 (2025)
15. Heuer, H., Wühr, P.: The impact of speed-accuracy instructions on spatial congruency effects. Journal of Cognition **6**(1), 49 (2023)
16. Jing, D., Wang, G., Liu, J., Tang, W., Sun, Z., Yao, Y., Wei, Z., Liu, Y., Lu, Z., Ding, M.: Mixture of horizons in action chunking. arXiv preprint arXiv:2511.19433 (2025)
17. Khayam, S.A.: The discrete cosine transform (dct): theory and application. Michigan State University **114**(1), 31 (2003)
18. Kim, B., Pahk, J., Lee, C., Kim, J., Lee, J., Kim, T.T., Shim, K., Lee, J.K., Zhang, B.T.: Espada: Execution speedup via semantics aware demonstration data downsampling for imitation learning. arXiv preprint arXiv:2512.07371 (2025)

19. Lan, Z., Mao, W., Li, H., Wang, L., Wang, T., Fan, H., Yoshie, O.: Bfa: Best-feature-aware fusion for multi-view fine-grained manipulation. IEEE Robotics and Automation Letters (2025)
20. Li, K., Malik, J.: Implicit maximum likelihood estimation. arXiv preprint arXiv:1809.09087 (2018)
21. Li, S., Gao, Y., Sadigh, D., Song, S.: Unified Video Action Model. In: Proceedings of Robotics: Science and Systems (June 2025). https://doi.org/10.15607/RSS.2025.XXI.074
22. Li, T., He, K.: Back to basics: Let denoising generative models denoise. arXiv preprint arXiv:2511.13720 (2025)
23. Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., Stone, P.: Libero: Benchmarking knowledge transfer for lifelong robot learning. Advances in Neural Information Processing Systems **36**, 44776–44791 (2023)
24. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: Rdt-1b: A diffusion foundation model for bimanual manipulation. In: International Conference on Learning Representations (2025)
25. Liu, Y., Hamid, J.I., Xie, A., Lee, Y., Du, M., Finn, C.: Bidirectional decoding: Improving action chunking via guided test-time sampling. In: International Conference on Learning Representations (2025)
26. Masuya, N., Sakaino, S., Tsuji, T.: Variable-frequency imitation learning for variable-speed motion. In: 2025 IEEE International Conference on Mechatronics (ICM). pp. 1–6. IEEE (2025)
27. Nam, T., Hwang, S.J.: Speedaug: Policy acceleration via tempo-enriched policy and rl fine-tuning. arXiv preprint arXiv:2512.00062 (2025)
28. Peternel, L., Sigaud, O., Babič, J.: Unifying speed-accuracy trade-off and cost-benefit trade-off in human reaching movements. Frontiers in human neuroscience **11**, 615 (2017)
29. Pierrieau, E., Dussard, C., Plantey-Veux, A., Guerrini, C., Lau, B., Pillette, L., George, N., Jeunet-Kelway, C.: Changes in cortical beta power predict motor control flexibility, not vigor. Communications Biology **8**(1), 1041 (2025)
30. Qi, H., Yin, H., Zhu, A., Du, Y., Yang, H.: Inference-time enhancement of generative robot policies via predictive world modeling. IEEE Robotics and Automation Letters **11**(5), 5534–5541 (2026). https://doi.org/10.1109/LRA.2026.3673995
31. Römer, R., Kobras, A., Worbis, L., Schoellig, A.: Failure prediction at runtime for generative robot policies. In: Advances in Neural Information Processing Systems. vol. 38, pp. 7631–7670 (2025)
32. So, J., Lee, C., Lee, S., Ok, J., Park, E.: Improving generative behavior cloning via self-guidance and adaptive chunking. arXiv preprint arXiv:2510.12392 (2025)
33. Suomalainen, M., Karayiannidis, Y., Kyrki, V.: A survey of robot manipulation in contact. Robotics and Autonomous Systems **156**, 104224 (2022)
34. Wang, H., Zhang, G., Yan, Y., Kompella, R.R., Liu, G.: Vla knows its limits. arXiv preprint arXiv:2602.21445 (2026)
35. Weng, Y., Zhang, X., Mu, Y., Zhu, Y., Li, Y., Liu, Q.: Temporal action selection for action chunking. arXiv preprint arXiv:2511.04421 (2025)
36. Xie, J., Wang, Z., Tan, J., Lin, H., Jiang, Y., Ma, X.: Subconscious robotic imitation learning. In: Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025). pp. 3559–3566 (2025)
37. Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., Levine, S.: Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In: Conference on robot learning. pp. 1094–1100. PMLR (2020)

38. Zhang, Q., Liu, Z., Fan, H., Liu, G., Zeng, B., Liu, S.: Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 14754–14762 (2025)
39. Zhang, W., Liu, H., Qi, Z., Wang, Y., Yu, X., Zhang, J., Dong, R., He, J., Wang, H., Zhang, Z., Yi, L., Zeng, W., Jin, X.: Dreamvla: A vision-language-action model dreamed with comprehensive world knowledge. In: Advances in Neural Information Processing Systems. vol. 38, pp. 24195–24228 (2025)
40. Zhao, T.Z., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware. In: Proceedings of Robotics: Science and Systems (Jul 2023). https://doi.org/10.15607/RSS.2023.XIX.016
41. Zhong, Y., Liu, Y., Xiao, C., Yang, Z., Wang, Y., Zhu, Y., Shi, Y., Sun, Y., Zhu, X., Ma, Y.: Freqpolicy: Frequency autoregressive visuomotor policy with continuous tokens. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
42. Zhu, M., Zhu, Y., Li, J., Wen, J., Xu, Z., Liu, N., Cheng, R., Shen, C., Peng, Y., Feng, F., et al.: Scaling diffusion policy in transformer to 1 billion parameters for robotic manipulation. In: 2025 IEEE International Conference on Robotics and Automation (ICRA). pp. 10838–10845. IEEE (2025)