STABLE-FAST: STABILIZING INFERENCE OF AUTOREGRESSIVE VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive Vision-Language-Action (VLA) models are a promising path toward generalist robot policies, yet their performance is critically dependent on action tokenization. The pioneering FAST tokenizer (Pertsch et al., 2025) is the first to enable autoregressive VLAs for dexterous, high-frequency tasks by compressing action sequences via the discrete cosine transform. However, this leads to *variable-length* token sequences for *fixed-length* action chunks, which causes cascading errors, jerky motions, and reduced task success. We introduce Stable-FAST, a tokenization strategy that resolves this instability at its source by partitioning action trajectories in the training dataset into *variable-length* action chunks that yield token sequences with markedly reduced length variance. This simple but effective reframing enables the training of VLAs with stabilized inference and markedly reduced action instability. Extensive experiments on real robots, using multiple VLA backbones including the π_0-FAST architecture, demonstrate that Stable-FAST improves action smoothness and increases the absolute task success rate by 10.8% on average, while reducing task completion times by over 40%. This offers a more reliable foundation for deploying autoregressive VLAs in the real world. We refer to the project page for the code and videos.

1 INTRODUCTION

The pursuit of generalizable robotic control has been profoundly influenced by the success of data-driven approaches in the broader machine learning community (Brown et al., 2020). A leading paradigm is to scale up models and data to produce generalist robot policies, capable of handling diverse tasks and generalizing to novel scenarios (Brohan et al., 2022; Ghosh et al., 2024). Vision-Language-Action (VLA) models, particularly those built on high-capacity Transformers, have emerged as a powerful architecture for this purpose (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024). Among VLA variants, autoregressive models (Zitkovich et al., 2023; Kim et al., 2024; Belkhale & Sadigh, 2024; Pertsch et al., 2025) are especially promising; their architectural and objective-level alignment with Large Language Models (LLMs) allows them to more readily benefit from the broad knowledge of pretrained models (Pertsch et al., 2025; Black et al., 2025). However, the efficacy of these models hinges on a critical component: the tokenization of continuous robot actions. This crucial step, which translates high-frequency motor commands into a discrete sequence for the Transformer, directly dictates the model's performance, efficiency, and the smoothness of the resulting robot behavior.

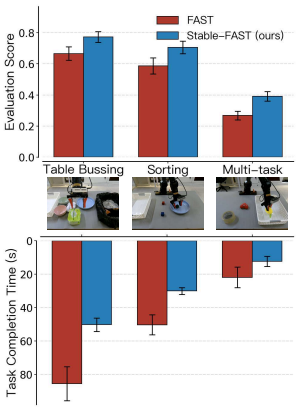

Figure 1: We propose Stable-FAST, a simple but effective approach to improve action tokenization for autoregressive VLA models. Stable-FAST enables much faster task completion (averaged only over successful trajectories) and better task performance.

A promising recent approach to action tokenization is the Frequency-space Action Sequence Tokenization (FAST) (Pertsch et al., 2025). By transforming continuous action chunks into the frequency domain via the discrete cosine transform (DCT), FAST provides a compressed and effective representation that enables autoregressive VLAs to learn high-frequency, dexterous skills where prior methods based on simple binning would fail. Despite its advantages, we identify a critical and

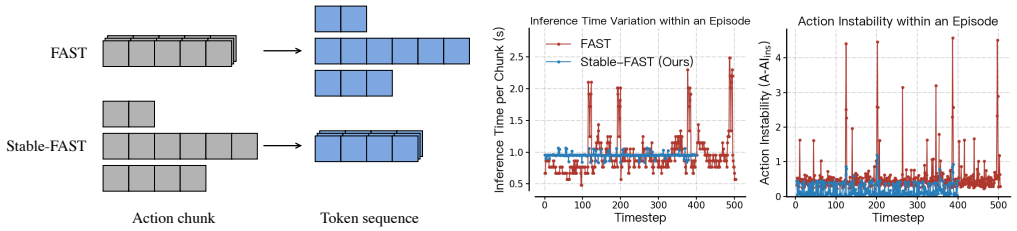

Figure 2: **By standardizing the complexity of action chunks, Stable-FAST stabilizes the length of tokenized sequences and reduces inference instability. Left:** Conceptual illustration. FAST maps fixed-length action chunks to token sequences of variable length. Our Stable-FAST inverts this, creating sequences with low-variance length from variable-length chunks. **Right:** The two resulting forms of instability. FAST's variable tokenization leads to high variance in inference time (left plot) and significant physical action instability (right plot), causing pauses and jerky motions. By ensuring a consistent prediction task, Stable-FAST resolves both issues, yielding stable latency and smooth actions. Plots show representative successful inference trajectories on the LIBERO benchmark (Liu et al., 2023).

unaddressed limitation in its formulation. We empirically find that the length of the token sequence generated by FAST is a direct proxy for the underlying **action complexity** of the chunk; simple motions like moving through free space yield short sequences, while intricate manipulations produce much longer ones. Consequently, FAST must map fixed-length chunks to variable-length tokens, which we found is the root cause of the instability we observe: it leads to unpredictable inference latency, a cascade of reconstruction and prediction errors, and ultimately, non-smooth actions that degrade task success, as illustrated in Figure 2 and detailed in Section 3.

In this paper, we introduce Stable-FAST, a novel tokenization strategy that directly addresses this fundamental challenge. Our key insight is to reframe the problem: instead of predicting fixed-length chunks of variable complexity, the VLA should learn to predict variable-length chunks of *fixed complexity* (Figure 2). We achieve this with a simple yet effective pre-processing method that dynamically adjusts each action chunk's length, using the number of non-zero DCT coefficients as a principled proxy for its complexity. This ensures that the resulting token sequences have small variance in length. Our contributions are threefold: (1) We are the first to formally identify and analyze how variable token lengths leads to unstable inference and degrades VLA performance. (2) We propose Stable-FAST, a practical tokenization strategy that stabilizes the token length of action chunks, leading to smoother and more effective policies. (3) We conduct extensive experiments across simulation and real-world settings, demonstrating that our approach significantly improves task success rates, action smoothness, and reduces task completion times (see Figure 1).

2 Preliminaries

In this section, we establish the technical foundation necessary to understand our work. We begin by framing the robot learning problem as an autoregressive sequence modeling task, and then provide a detailed overview of the FAST tokenizer, the state-of-the-art method that our work builds upon and aims to improve. Finally, we introduce the key metrics used to quantitatively measure action instability, which is the central problem we address.

Problem formulation. We formulate the robot learning problem as a sequence modeling task. At each timestep t, the robot receives a visual observation o_t and a natural language instruction l. The goal is to learn a policy $\pi(a_{1:H}|o_t, l)$ that generates a action chunk of length H, which makes it easier to produce temporally-consistent actions and reduces compounding error. Typically, the number of actions H is fixed throughout training and inference. The goal of action tokenization is to define two mappings—a tokenizer $\mathcal{T}_E : a_{1:H} \to T_{1:N}$ from an action chunk $a_{1:H}$ to a sequence of discrete tokens $T_{1:N}$ with vocabulary size $|\mathcal{V}|$, and a detokenizer $\mathcal{T}_D : T_{1:N} \to a_{1:H}$ that maps back from tokens to actions. With this formulation, an autoregressive VLA policy with parameters θ is trained to model the conditional probability of the token sequence $T_{1:N}$ given current observation o_t and language instruction l in an autoregressive manner:

$$\log p(T_{1:N}|o_t, l) = \sum_{n=1}^{N} \log p_\theta(T_n|T_{<n}, o_t, l). \tag{1}$$

During inference, the policy generates the token sequence token by token, and the full sequence is then passed to the detokenizer $\mathcal{T}_D$ to produce a continuous action chunk executed by the robot. In this framework, the length of the token sequence, N, becomes a critical variable influencing both inference speed and model performance, which is the central focus of our work.

The FAST tokenizer. The core idea of the FAST tokenizer is to first transform an action chunk $a_{1:H} \in \mathbb{R}^{H \times D}$ from the time domain to the frequency domain using the discrete cosine transform (DCT). For each action dimension $d \in \{1, \ldots, D\}$, this produces a vector of H frequency coefficients. The resulting coefficients for all dimensions are then concatenated and flattened into a single vector of length $H \times D$. This vector is subsequently tokenized into a sequence of discrete tokens $T_{1:N}$ using a standard text tokenization algorithm, such as Byte Pair Encoding (BPE) (Gage, 1994). Typically, N is much smaller than $H \times D$, thus producing a compressed representation. The corresponding detokenizer, $\mathcal{T}_D$, first decodes the token sequence $T_{1:N}$ back into the flattened vector of frequency coefficients using BPE. This vector is then reshaped back to the size $H \times D$, and the inverse discrete cosine transform is applied to each dimension to reconstruct the action chunk $a_{1:H}$. A key characteristic of the whole process is that for a fixed action chunk length H, the number of tokens N generated by the BPE algorithm is *variable*, depending on the complexity of the action chunk itself (as will be demonstrated in Section 3).

Metrics for action smoothness. To quantitatively analyze the smoothness and dynamic properties of action trajectories, we adopt two metrics from recent work (Valle et al., 2025): one based on action velocity and another on action acceleration. These metrics are derived from the finite differences of the action sequence. The first metric, Action Velocity Metric (A-VI), is the second-order difference of the action sequence. This corresponds to the discrete second derivative (acceleration) and is effective at capturing sharp changes in velocity. For a trajectory τ including N actions $a_{1:N}$, it is calculated as:

$$\text{A-VI}(\tau) = \frac{1}{N-2} \sum_{t=3}^{N} ||\Delta a_t - \Delta a_{t-1}||_1 = \frac{1}{N-2} \sum_{t=3}^{N} ||a_t - 2a_{t-1} + a_{t-2}||_1. \quad (2)$$

The second metric, Action Acceleration Metric (A-AI), further captures fine-grained, jerky motions by measuring the rate of change in acceleration. It is defined as the third-order difference of the action sequence, which corresponds to the discrete third derivative (jerk):

$$\text{A-AI}(\tau) = \frac{1}{N-3} \sum_{t=4}^{N} ||\Delta^2 a_t - \Delta^2 a_{t-1}||_1 = \frac{1}{N-3} \sum_{t=4}^{N} ||a_t - 3a_{t-1} + 3a_{t-2} - a_{t-3}||_1. \quad (3)$$

In general, higher values for these metrics indicate a less smooth and more dynamic action sequence. In Section 4, we will show how these fundamental metrics can be repurposed to define both the intrinsic complexity of a task and the erroneous instability of a policy.

3 From Variable Token Length to Action Instability

In this section, we conduct a detailed analysis to empirically validate our central hypothesis: that the variable action complexity inherent in robotic tasks is the primary source of inference instability and performance degradation when using the FAST tokenizer. We utilize the standard LIBERO dataset (Liu et al., 2023) for this analysis, as it encompasses a wide range of manipulation behaviors with varying action complexities.

3.1 Disentangling Complexity from Instability: A More Rigorous Metric

To rigorously test our hypothesis, we must first precisely disentangle two related but critically different concepts: the intrinsic *complexity* of a task and the erroneous *instability* of a policy. Complexity describes the necessary, and often highly dynamic, motion an expert must perform to succeed; it is an objective, neutral property of the task itself. Instability, in contrast, is an undesirable artifact of the policy—it is the unnecessary, oscillatory motion produced by control errors. A simple measurement of a policy's output conflates these two: the observed motion is a combination of the task's required complexity and the policy's own instability. Therefore, to truly understand and quantify policy failure, we must isolate the latter from the former. In the following, we provide formal definitions for these concepts to achieve this separation.

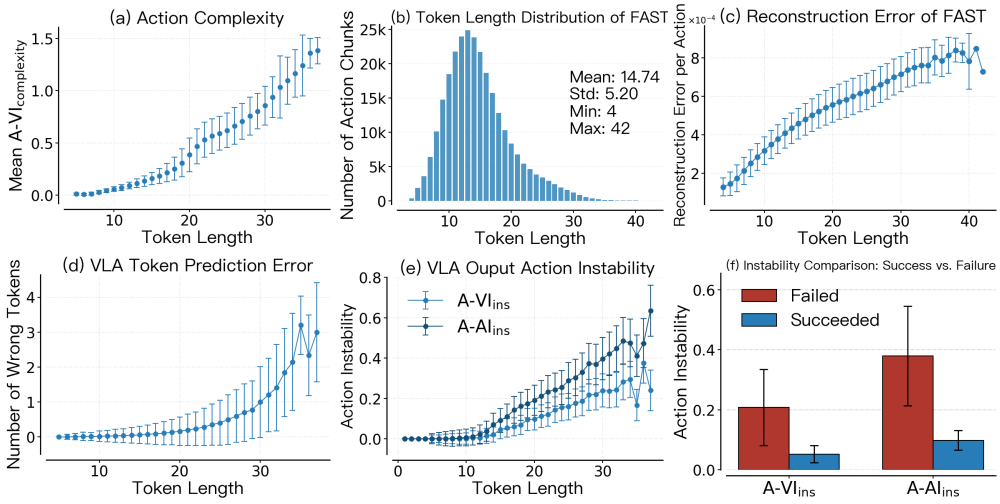

Figure 3: **Analyzing the mechanism of instability from variable action complexity to task failure. (a)** The token sequence length generated by FAST for a fixed-length chunk shows a strong positive correlation with the chunk's intrinsic action complexity. **(b)** The wide distribution of token lengths thus reflects the wide range of action complexities present in the dataset. **(c)** The tokenizer's reconstruction fidelity degrades for longer sequences, as these correspond to more complex actions that are harder to accurately compress. **(d)** The VLA's token prediction error rises sharply for these longer, more complex sequences, which represent an intrinsically harder prediction target. **(e)** This cascade of errors manifests as action instability in the robot's output, representing excess, erroneous motion. **(f)** Critically, this instability is directly correlated with task outcomes, with failed episodes exhibiting significantly higher levels of erroneous motion than successful ones. Together, these results establish an unbroken causal chain, beginning with the initial variable action complexity and culminating in task failure.

Action complexity. We begin by defining the intrinsic difficulty of a task. Action complexity is an objective and neutral descriptor of the physical motion required by an expert to solve a task. To quantify this concept, we employ the A-VI and A-VI metric (as defined in Section 2), applying it to the expert demonstration data. For a given task l within a dataset of demonstrations $\mathcal{D}$, its action complexity wrt A-VI is formalized as the expected A-VI over all expert trajectories:

$$\text{A-VI}_{\text{complexity}}(l, \mathcal{D}) = \mathbb{E}_{\tau \sim \mathcal{D}(l)}[\text{A-VI}(\tau)], \tag{4}$$

where $\mathcal{D}(l)$ are the subset of demonstrations $\mathcal{D}$ of task l. Action complexity wrt A-AI is similarly defined. This value represents the baseline level of necessary action dynamics inherent to task l.

Action instability. In contrast, action instability is an undesirable artifact of the policy, not the task. To quantify this, we must isolate this erroneous motion from the necessary motion defined above. We first measure the total dynamism of a model π's behavior by taking the expected A-VI over its inference trajectories ($\tau' \sim \pi(l)$). Our core insight is that the true instability is the portion of this motion that exceeds the task's intrinsic complexity:

$$\text{A-VI}_{\text{ins}}(l, \pi, \mathcal{D}) = \mathbb{E}_{\tau' \sim \pi(l)}\left[\text{A-VI}(\tau')\right] - \text{A-VI}_{\text{complexity}}(l, \mathcal{D}). \tag{5}$$

A-AI$_{\text{ins}}$ can be similarly defined. This metric isolates the purely erroneous component of the motion. A value near zero implies the model is as smooth as the expert, while a high positive value indicates significant unnecessary motion. For the LIBERO benchmark, this metric is directly computable. All subsequent mentions of "instability" in our analysis refer to this instability metric.

3.2 Empirical Analysis of the Instability Cascade

Having established our precise metrics, we now proceed with the empirical investigation of our central hypothesis, illustrated in Figure 3. Our analysis begins by validating the foundational premise of our work: that token length in the FAST framework is a direct proxy for action complexity. Figure 3(a) plots the token sequence length against action complexity of the corresponding chunk, measured by A-VI$_{\text{complexity}}$. The strong, near-monotonic positive correlation provides clear empirical evidence for this link.

With this relationship established, the wide, right-skewed distribution of token lengths shown in Figure 3(b) can be understood as a direct reflection of the diverse action complexities present in the LIBERO dataset. With a mean of 14.74 tokens and a standard deviation of 5.20, it is evident that action chunks of the same length can represent an order-of-magnitude difference in underlying complexity. This variability in action complexity is the starting point of the instability cascade that we investigate next.

Next, we examine how this variability impacts the tokenizer's own performance. As shown in Figure 3(c), this variability is not benign; it is directly correlated with a decrease in the tokenizer's reconstruction fidelity. Longer token sequences generally correspond to more complex, high-frequency actions that have more significant, non-zero DCT coefficients. The quantization and rounding steps in FAST introduce more error for these complex signals, and the BPE algorithm requires more tokens to represent them. The plot shows that the mean reconstruction error, normalized per action in the chunk, increases steadily with the token sequence length. This indicates that the action chunks requiring longer token sequences are also the ones that are reconstructed less accurately, introducing an initial source of error before the policy even makes a prediction.

The challenge is further compounded during policy learning, as the VLA must learn to predict these variable-length sequences. Figure 3(d) reveals that the VLA's token prediction error rises sharply as the target sequence length increases. This demonstrates a non-linear, almost exponential trend, where longer sequences become significantly more difficult for the model to predict accurately. This difficulty arises from two sources: first, the autoregressive nature of the model means that the probability of an error compounds with each additional token that must be predicted. Second, as established, longer sequences represent more intricate and complex actions, which are intrinsically harder for the policy to infer from a given visual observation and language instruction.

Ultimately, these compounding errors manifest as physical instability in the robot's behavior. Figure 3(e) demonstrates a strong positive correlation between the generated token length and the action instability of the VLA's output, as measured by both A-VI_{ins} and A-AI_{ins}. This shows that the inaccurate predictions for longer token sequences, which themselves have higher reconstruction error, are detokenized into jerky and oscillatory action trajectories. Figure 3(f) provides the final, critical piece of evidence, directly linking this physical instability to task outcomes. Across both A-VI_{ins} and A-AI_{ins} metrics, episodes that ended in failure consistently exhibit significantly higher levels of instability compared to those that succeeded.

This analysis, therefore, validates our central hypothesis by establishing a clear chain of consequences: from the variable action complexity inherent in the task, to the variable token lengths produced by FAST, to the cascading reconstruction and prediction errors, to physically unstable actions, and finally, to a tangible increase in task failure. Having identified and empirically verified this fundamental issue, we are motivated to develop a solution that directly targets it.

4 Standardizing Action Complexity for Stable Tokenization

Our analysis pinpoints the source of instability in the FAST tokenization scheme to its variable-length token output, a direct result of processing fixed-length chunks of varying complexity. To build a robust VLA policy, we must first stabilize the very foundation it is built upon by standardizing the complexity of the action chunks.

Our approach inverts the standard paradigm: rather than processing action chunks of fixed length and variable complexity, we adaptively segment the action trajectory at training into chunks of variable length to ensure each possesses a consistent level of complexity. This principle is the cornerstone of Stable-FAST, our novel tokenization paradigm. By adopting this approach, we shift the problem from managing variable-length outputs to a new challenge in data preprocessing: how do we partition the raw action trajectories to create these iso-complex chunks? We can no longer use a simple, fixed-size sliding window; instead, we need a principled strategy to determine the appropriate length, H, for each chunk.

To implement this strategy, we require a principled, low-cost proxy for action complexity that is computable *before* the BPE tokenizer is trained. The theory of DCT provides a natural and computationally efficient solution. Due to its energy compaction property (Khayam, 2003), the DCT concentrates the informational content of a smooth, simple signal into a few sig-

nificant low-frequency coefficients. Conversely, a complex, high-frequency signal requires a larger number of non-zero coefficients for an accurate representation. The number of *non-zero DCT coefficients* therefore serves as a theoretically grounded measure of an action chunk's intrinsic complexity. As empirically verified in the top plot of Figure 4, this choice is validated by a near-perfect linear correlation between the non-zero coefficient count and the final number of BPE tokens, confirming it can serve as a highly effective and reliable proxy.

An alternative might be to use the token count from the FAST tokenizer to decide the horizon H for each chunk. However, this approach introduces a problematic circular dependency: the data partitioning would depend on a specific tokenizer, but we must then train a *new* tokenizer on this new partition. This process is inherently unstable and can cause the tokenizer's properties to drift unpredictably. Our experiments confirm this instability; using this method resulted in a significant shift in the tokenizer's characteristics, increasing the average token count from 14.74 to 20.62 (shown in Figure 4) and the average action chunk length from 10 to 13.08. In contrast, our non-zero coefficient approach is independent of any specific BPE vocabulary, providing a stable and principled foundation.

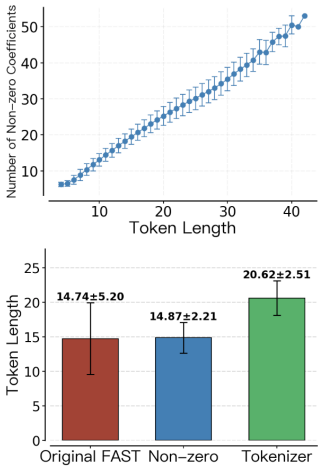

Figure 4: **Top:** The number of non-zero DCT coefficients shows a near-perfect linear correlation with the final token length. **Bottom:** Using this proxy successfully reduces the token length's standard deviation while maintaining the mean. An unstable alternative ("Tokenizer") is shown to fail, causing the mean to drift significantly.

Our complete data preparation and training pipeline is formalized in Algorithm 1 and proceeds as follows. First, we establish a target complexity, C_{target}, by calculating the average number of non-zero DCT coefficients over action chunks of a fixed, reasonable length (e.g., $H = 10$) across the entire training dataset. Second, we reprocess the trajectories to generate a new dataset of variable-length action chunks, each with a complexity that closely matches this target. For each potential starting point in a given trajectory, we iteratively expand an action chunk by incrementing its length H. If the chunk boundary exceeds the trajectory's length, we pad the chunk by repeating the final action. The expansion continues until the chunk's complexity just surpasses C_{target}. We then select the chunk, of length H or $H - 1$, whose complexity is numerically closer to C_{target}, thereby creating a new dataset, $\mathcal{D}_{\text{stable}}$, where each entry has a near-constant complexity. Finally, we train a new BPE tokenizer on this dataset of variable-length chunks and then use it to train the VLA policy. The VLA model, trained on these new sequences, implicitly learns to predict the appropriate action chunk length by generating token sequences of low-variance length.

As shown in the bottom plot of Figure 4, our Stable-FAST approach is highly effective. The standard deviation of the token sequence length is markedly reduced from 5.20 to 2.21. This stabilization is achieved without sacrificing efficiency; the mean token count remains remarkably consistent (14.74 for FAST vs. 14.87 for Stable-FAST), as does the average action chunk length (10.0 for FAST vs. 11.4 for Stable-FAST). This demonstrates that our method successfully standardized the complexity of the prediction task, evidenced by the stabilized token length, thus laying a more principled and stable foundation for training and deploying autoregressive VLA models.

5 Experiments

This section validates Stable-FAST in both simulation and real-world environments. We demonstrate that our core approach of standardizing action complexity yields policies with markedly improved success rates, action smoothness, and task completion time.

5.1 Experimental Setup

Evaluation tasks. As illustrated in Figure 5, we test VLA performance on 4 evaluation tasks (3 real robot, 1 simulated), ranging from simple manipulation tasks to complex, long-horizon tasks.

- **LIBERO**: Following Pertsch et al. (2025), we train a single policy on a mixture of 40 tasks from the 4 splits of the LIBERO benchmark (LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10), and measure average performance across them.
- **Table bussing**: A complex, long-horizon manipulation task requiring the robot to place all trash items into a trash bin and all plates and bowls into a plastic container. The task requires generalization across various combinations and spatial arrangements of objects.
- **Sorting**: A task demanding precise, sequential manipulation, where the robot must sort small wooden blocks of different colors into their designated containers (a white basket or a blue plate) in the correct order. The key challenges lie in adhering to the sequence and ensuring accurate placement into the appropriate container.
- **Multi-task**: Pick-and-place tasks designed to test language following capabilities in the presence of distractors. The task consists of 10 distinct objects (e.g., lemon, bread, bowl). The policy is trained jointly on all objects, and during evaluation, random distractors are introduced.

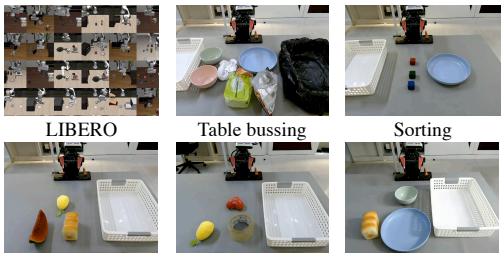

Figure 5: **Evaluation environments.** We evaluate Stable-FAST on a diverse suite of 1 simulation and 3 real-world tasks. These tasks are designed to test VLA performance across a spectrum of complexity, ranging from simple manipulation tasks to complex, long-horizon tasks.

Implementation details. All real-world experiments are conducted using an AgileX Piper robotic arm equipped with a wrist-mounted and a third-person camera. For our dataset, we collected 50 expert trajectories for both the Table Bussing and Sorting tasks. For the Multi-task setup, we gathered 30 demonstrations for each of its 10 sub-tasks, yielding a total of 300 expert trajectories. We employ a consistent pipeline to train all policies. Each model is trained for 320000 steps using a cosine annealing learning rate schedule, which is initialized at 2×10^{-5} with a 1000-step warmup phase. For evaluation, we assess each of the 40 LIBERO simulation tasks over 50 trials, resulting in 2000 total rollouts. The Table bussing and Sorting tasks are each evaluated for 25 trials. The Multi-task setup is also evaluated with 25 trials for each of its 10 sub-tasks, totaling 250 real-world rollouts. To ensure metrics are not skewed by random failures, the A-VI_{ins}, A-AI_{ins}, and task completion time are calculated exclusively from successful trajectories. Except for the VLM backbone comparison in Figure 7, all our experiments use SmolVLM-500M (Marafioti et al., 2025) to initialize the VLA for training.

5.2 Ablation Studies

We conduct a comprehensive ablation study to analyze three key aspects of our method on LIBERO: (1) We investigate how varying the target action complexity affects policy performance and compare our method against the FAST. (2) We assess the benefits of our method across different VLA backbones. (3) We position our contribution by comparing Stable-FAST against preceding tokenization paradigms—learned vector quantization (VQ) (Lee et al., 2024) and FAST—to demonstrate a clear progression in solving the trade-off between representational fidelity and inference stability.

In our first ablation study, we evaluate Stable-FAST by focusing on its key hyperparameter: the target action complexity. The goal is to understand how varying this parameter affects policy performance and to benchmark our method against the FAST tokenizer. To create a fair comparison, as FAST requires a fixed action chunk length, we designed two baselines. The first, FAST-C (for similar **C**omplexity), uses a fixed chunk length chosen such that the *average* complexity of its resulting tokens matches the target complexity level of Stable-FAST. The second, FAST-L (for similar chunk **L**ength), sets its fixed chunk size to match the *average* length of the variable-length chunks produced by Stable-FAST at a given complexity setting (see Appendix D for details).

Our primary finding, illustrated in Figure 6(a), is that Stable-FAST's complexity-driven partitioning mechanism is highly effective. It consistently and significantly reduces the standard deviation, σ, of the token sequence length across all evaluated complexity levels, directly validating its ability to standardize chunk complexity. This stabilization at the tokenization level has a direct and positive impact on the policy's behavior. As shown in Figures 6(b) and (c), the smoother and more consistent

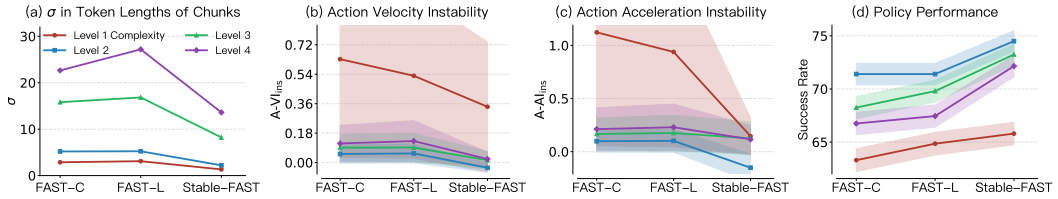

Figure 6: **Ablation study on target action complexity.** We evaluate Stable-FAST across four complexity levels and compare it against two FAST baselines that are calibrated to match Stable-FAST's properties at each level: FAST-C, which is matched for chunk complexity, and FAST-L, which is matched for average chunk length (See Appendix D for details). **(a)** Standard deviation of the output token sequence length per chunk. A lower value indicates more stable tokenization. **(b, c)** The resulting A-VI_{ins} and A-AI_{ins}, where lower is better. **(d)** Final task success rate, where higher is better. Stable-FAST demonstrates lower token length variance, which translates to reduced action instability and higher task success.

tokenization leads to a marked reduction in both action velocity instability (A-VI_{ins}) and action acceleration instability (A-AI_{ins}). This reduction in erroneous, jerky motion ultimately translates to higher task success rates, as shown in Figure 6(d).

The study also reveals that, analogous to chunk size in FAST, the target complexity is a crucial hyperparameter for Stable-FAST. Performance is not monotonic with complexity; instead, an optimal level exists. For the LIBERO benchmark, a "level 2" complexity yields the best results. Setting the complexity too low (level 1) results in overly short action chunks that can lead to temporally-inconsistent actions and compounding errors (Zhao et al., 2023; Pertsch et al., 2025). Conversely, setting it too high (levels 3, 4) causes even Stable-FAST to produce long token sequences with higher variance, leading back to the action instability and lower success rates we sought to avoid.

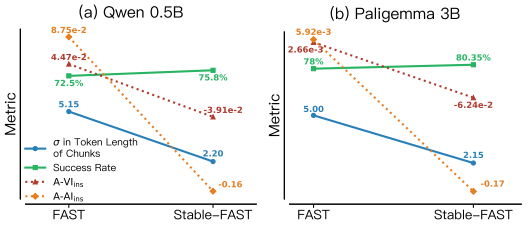

Figure 7: **Stable-FAST consistently excels across different VLMs.** We compare performance using two other backbones: **(a)** Qwen 0.5B, as used in MiniVLA, and **(b)** Paligemma 3B, the VLM used in π series. For both, replacing the FAST with Stable-FAST yields consistent improvements.

Remarkably, when operating at its optimal complexity level, Stable-FAST achieves negative values for both action instability metrics. By our definition of instability, this implies that the learned policy generates actions that are even smoother and more dynamically consistent than the expert demonstrations themselves. This suggests that the expert data contains a degree of suboptimal, unnecessary motion, and that the stabilizing effect of our method is powerful enough to filter out this inherent noise. This demonstrates that Stable-FAST not only stabilizes policy inference but can also enable the policy to learn behavior that surpasses the qualitative limitations of the training data.

In our second ablation, we test if the benefits of Stable-FAST generalize across different VLM backbones. We replaced our standard SmolVLM with both Qwen-0.5B (Bai et al., 2025) and PaliGemma 3B used in π model series (Black et al., 2024; 2025; Pertsch et al., 2025). The results in Figure 7 show that Stable-FAST consistently provides superiority over the FAST tokenizer, regardless of the underlying VLM. This confirms our method's benefits are fundamental and not architecture-specific. We also observe a clear trend: as the VLM's capability increases (PaliGemma $>$ SmolVLM $>$ Qwen), action instability decreases and success rates improve. This suggests that more powerful foundation models contribute significantly to learning effective control policies, and Stable-FAST provides a robust tokenization layer to unlock their potential.

Table 1: **Comparison with alternative tokenizer paradigms.** Stable-FAST shows superiority against both VQ method and FAST across all metrics.

	VQ	FAST	Stable-FAST
σ of Token Length (↓)	**0.00**	5.20	2.21
% Success Rate (↑)	66.45±1.06	71.40±1.01	**74.50** ±0.97
A-VI_{ins}($\times 10^{-2}$)(↓)	10.06±8.55	5.24±5.60	**-3.22**±2.74
A-AI_{ins}($\times 10^{-2}$)(↓)	15.64±13.68	9.84±10.27	**-15.10**±12.84

In our third ablation, we contextualize our contribution by positioning Stable-FAST within the evolution of action tokenization, comparing it against a learned VQ approach and the FAST. The results

Table 2: **Stable-FAST on real-world tasks.** σ denotes the standard deviation in the token lengths of generated action chunks. Stable-FAST outperforms FAST across all metrics.

		σ	Success Rate%	Task Progress%	A-VI$_{ins}$($\times 10^{-3}$)	A-AI$_{ins}$($\times 10^{-3}$)	Time (s)
Table Bussing	FAST	2.39	16.00±7.48	66.67±4.30	19.2±17.3	37.5±30.0	85.47±10.21
	Stable-FAST	1.60	24.00±8.72	77.33±3.45	-6.56±7.56	-15.9±13.4	50.23±3.99
Sorting	FAST	2.74	12.00±6.63	58.67 ±5.19	21.3±18.1	29.1±26.1	50.31±5.95
	Stable-FAST	1.85	24.00±8.72	70.67±4.00	-8.68±9.24	-22.8±10.3	30.09±2.12
Multitask	FAST	1.00	26.80±2.81	/	23.7±6.0	43.5±11.4	21.92±6.19
	Stable-FAST	0.77	39.20±3.09	/	-5.74±3.27	-38.9±7.8	12.40±3.00

in Table 1 reveal a clear performance hierarchy, showing that FAST comprehensively outperforms VQ, and Stable-FAST, in turn, outperforms FAST in both success rate and action stability. This demonstrates a clear and logical progression in tokenization methods.

This performance trend is explained by the evolution of the tokenization paradigm. Learned VQ-based tokenizers operate on a rigid "fixed-in, fixed-out" principle, creating a representational bottleneck that struggles with variable action complexity. FAST was a significant advancement, resolving this with a "fixed-in, variable-out" approach that allows the token representation to adapt to the action's complexity. However, this introduced a new challenge: the variable output complexity became a direct source of action instability. Stable-FAST resolves this final issue by inverting the process to a "variable-in, fixed-complexity-out" strategy. It not only retains the crucial representation flexibility of FAST but also stabilizes the inference of VLAs, marking the next logical step in developing robust action tokenizers.

5.3 Real-World Performance of Stable-FAST

To validate our approach in the complexities of a physical environment, we evaluate Stable-FAST on three real-world manipulation tasks, directly comparing Stable-FAST against the FAST tokenizer. The results, summarized in Table 2, provide compelling evidence that the benefits of our method hold true amidst the noise and randomness of the real world. Stable-FAST consistently produced token sequences with lower standard deviation, which directly translated to policies with markedly lower action instability. This enhanced motion smoothness and control, in turn, led to a superior task success rate across all evaluated tasks.

Crucially, these experiments revealed a significant system-level advantage: a reduction in task completion time by over 40%. We attribute this significant speed-up to two factors. First, the enhanced stability of the generated motions enables the policy to complete tasks more directly, requiring fewer steps overall (e.g., Figure 2). Second, in non-blocking control systems where action generation and execution overlap, the variable-length outputs of FAST can force the robot to pause while the VLA processes a complex action. In contrast, Stable-FAST's consistent prediction complexity facilitates a synchronized, rhythmic pipeline between the VLA and the robot, minimizing idle time. This underscores that stabilizing inference is critical not just for success, but for creating a more efficient and practical robotic system.

6 Conclusion

In this work, we introduced Stable-FAST, a novel action tokenization strategy designed to address the critical issue of inference instability in autoregressive VLA models. While the pioneering FAST tokenizer enabled high-frequency control, its variable-length token outputs introduced unpredictable variance into the model's prediction target, leading to erratic actions. We resolve this by re-framing the tokenization problem: instead of encoding fixed-length action chunks into sequences of variable complexity, we encode variable-length chunks into sequences of standardized complexity. This fundamental shift allows Stable-FAST to generate token sequences with significantly reduced length variance. Both simulated and real-world experiments demonstrate that Stable-FAST enables the training of VLAs with markedly improved inference reliability, smoother physical actions, and higher task success rates, providing a more robust foundation for training and deploying the high-performance, generalist robot policies.

REFERENCES

Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL `https://github.com/Stanford-ILIAD/openvla-mini`.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4788–4795. IEEE, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595*, 2023.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Huang Huang, Fangchen Liu, Letian Fu, Tingfan Wu, Mustafa Mukadam, Jitendra Malik, Ken Goldberg, and Pieter Abbeel. Otter: A vision-language-action model with text-aware visual feature extraction. *arXiv preprint arXiv:2503.03734*, 2025.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. *arXiv preprint arXiv:2410.24185*, 2024.

Syed Ali Khayam. The discrete cosine transform (dct): theory and application. *Michigan State University*, 114(1):31, 2003.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37:4062–4089, 2024.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya 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)*, pp. 6892–6903. IEEE, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Pablo Valle, Chengjie Lu, Shaukat Ali, and Aitor Arrieta. Evaluating uncertainty and quality of visual language action-enabled robots. *arXiv preprint arXiv:2507.17049*, 2025.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

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

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A THE USE OF LARGE LANGUAGE MODELS

In the preparation of this paper, we employed Large Language Models (LLMs) solely as a writing assistance tool for limited text polishing and language refinement. LLMs were not involved in any aspects of research ideation, conceptual development, technical analysis, algorithm design, experimental execution, or result analyses. All scientific contributions, methodological innovations, and intellectual content remain entirely our own.

B RELATED WORK

Vision-Language-Action models. The paradigm of learning robotic policies from large, diverse datasets (O'Neill et al., 2024; Fang et al., 2023; Walke et al., 2023; Khazatsky et al., 2024; Bharadhwaj et al., 2024; Jiang et al., 2024; Bu et al., 2025) has gained significant traction, inspired by the tremendous success of foundation models in natural language and vision. This has led to the development of VLA models (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Pertsch et al., 2025; Zawalski et al., 2024; Cheng et al., 2024; Belkhale et al., 2024; Abeyruwan et al., 2025; Bjorck et al., 2025; Black et al., 2025; Shi et al., 2025; Huang et al., 2025; Belkhale & Sadigh, 2024; Kim et al., 2025), which leverage high-capacity architectures, typically Transformers, to map raw sensory inputs (vision) and high-level instructions (language) directly to robot actions. Different generative modeling approaches for continuous actions have been explored. One way is to use diffusion models (Ghosh et al., 2024; Bjorck et al., 2025) or flow-based models (Black et al., 2024; 2025), which can directly model complex, continuous action distributions. An alternative and equally powerful way, and the focus of our work, is autoregressive modeling. Autoregressive VLAs (Zitkovich et al., 2023; Kim et al., 2024; Belkhale & Sadigh, 2024; Pertsch et al., 2025) frame robot control as a sequential prediction problem, generating action tokens one by one conditioned on visual and language context. This formulation offers a distinct advantage in its seamless architectural alignment with Large Language Models (LLMs), facilitating knowledge transfer from web-scale pretrained models.

Action representations of autoregressive VLA models. The discretization of continuous, high-dimensional action spaces is a pivotal design choice in autoregressive VLAs. The most straightforward and widely adopted method is uniform per-dimension binning, where each dimension of the action vector at each timestep is independently discretized into a set of uniform bins (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2024). However, as control frequency or action dimensionality increases, it leads to excessively long token sequences, which makes inference time prohibitively long and poses difficulties in modeling fine-grained, dynamic movements due to high similarities between consecutive actions (Pertsch et al., 2025). To address these limitations, subsequent research has explored more compressed action representations, including learned vector-quantized action representations (Lee et al., 2024; Belkhale & Sadigh, 2024; Mete et al., 2024). By encoding short action chunks into discrete latent codes, these methods can create a vocabulary of reusable action "primitives", leading to more compact sequences. However, this approach fails on high-frequency tasks when fine-grained control is required (Pertsch et al., 2025). Most recently, Pertsch et al. (2025) introduced the FAST tokenizer. By transforming action chunks into the frequency domain using DCT, it provides a highly efficient and effective representation that captures both low-frequency trends and high-frequency details of an action trajectory, enabling autoregressive VLAs to excel at dexterous, high-frequency tasks where prior methods struggled. Our work directly builds on the FAST framework. While we retain its powerful frequency-domain representation, we diagnose and resolve a fundamental instability in its tokenization process.

C ADDITIONAL EXPERIMENTAL DETAILS

This section provides further details on the experimental setup and data analysis procedures for the results presented in the main paper.

C.1 HARDWARE AND ENVIRONMENT

All real-world experiments in Section 5 were conducted on a single workstation equipped with an NVIDIA GeForce RTX 4090D GPU. Other inference experiments, including data analysis in

Section 1 and Section 3, were conducted on an NVIDIA GeForce RTX 3090 GPU. All real-world demonstrations were collected at a frequency of 10 Hz.

C.2 Analysis Details for Figure 2

The two plots on the right of Figure 2, illustrating the inference time variation and action instability within an episode, were generated from the same single, representative inference trajectory. This was done to qualitatively showcase the direct correlation between the moments of high inference latency and the resulting physical action instability during a single run.

C.3 Data Analysis Protocol for Figure 3

The comprehensive analysis in Figure 3, which traces the causal chain from action complexity to task failure, was conducted as follows:

- **Subplots (a), (b), and (c):** The data for these plots, which analyze action complexity, the token length distribution of FAST, and the reconstruction error of FAST, were derived by processing the entire training dataset of all 40 LIBERO tasks. These statistics reflect the intrinsic properties of the dataset and the FAST tokenizer itself, independent of a trained policy.
- **Subplots (d) and (e):** The results for VLA token prediction error and VLA output action instability were obtained using a VLA policy that was trained with the FAST tokenizer and initialized with SmolVLM weights. To generate the data, we performed single-step inference on every observation contained within the LIBERO training set. The model's predicted tokens and the resulting detokenized actions were then compared against the corresponding ground-truth tokens and actions from the dataset to calculate the error and instability metrics.
- **Subplot (f):** The "instability comparison: success vs. failure" plot was also generated using the same VLA model (FAST tokenizer + SmolVLM). The statistics were compiled by analyzing full inference trajectories executed by the policy. These rollouts were obtained from the LIBERO environment, and the resulting action instability metrics were aggregated and conditioned on whether the episode ended in success or failure.

C.4 Multi-task Setup Details

The multi-task environment is designed to evaluate the policy's ability of language instruction following. This setup comprises 10 distinct pick-and-place sub-tasks, each involving a different object.

Data collection. We collected a total of 300 expert trajectories for this setup, with 30 demonstrations for each of the 10 sub-tasks. To enhance the policy's robustness, two random distractors are present in the scene during data collection for each demonstration. All data was recorded at a sampling frequency of 10Hz.

Task list. The policy was jointly trained on the following 10 sub-tasks, with the instruction format "pick up the [object] and place it into the basket". See Figure 8.

System configuration. (a) Inputs. The policy receives visual input from two cameras: a wrist camera and a third-person camera. Both provide images at a resolution of 640×480 pixels. Proprioceptive states from the robot arm are also included as input. (b) Action Execution. During deployment, the policy predicts an action chunk, and all actions within that chunk are executed sequentially by the robot.

D Derivation of the Four Complexity Levels for Ablation Study

We details the method used to establish the 4 complexity levels for the ablation study presented in Figure 6. Our goal is to create a rigorous comparison between Stable-FAST and FAST. To achieve this, we defined two calibrated baselines, FAST-C (matched for **C**omplexity) and FAST-L (matched

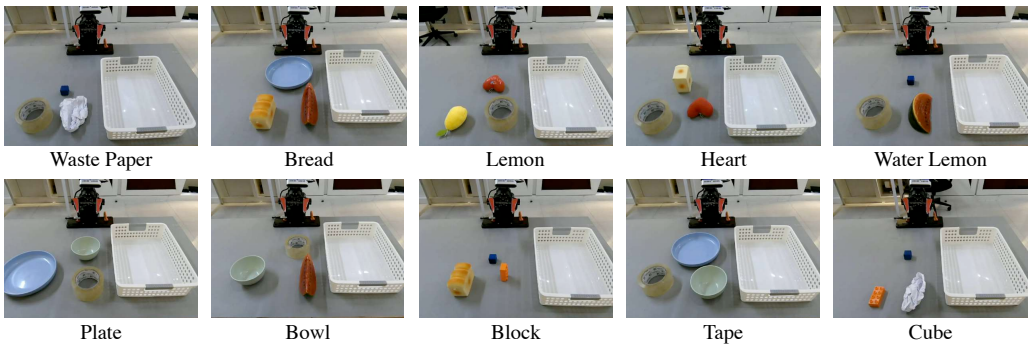

Figure 8: **Overview of the Multi-task sub-tasks.** The evaluation involves 10 distinct object manipulation tasks (paper, bread, lemon, heart, watermelon, plate, bowl, block, tape, and cube). Random distractors are present during both training and inference.

for average chunk **L**ength), for each complexity level. As the number of non-zero DCT coefficients serves as our proxy for complexity, we used its value to define these complexity levels. The calibration process proceeded in two main stages.

1. Calibrating target complexities and defining FAST-C. We first established the 4 target complexity values that guide the Stable-FAST data preparation pipeline. To do this, we began with the standard FAST framework and set 4 representative fixed chunk lengths: 5, 10, 30, and 50. For each of these lengths, we processed the entire LIBERO dataset and computed the average number of non-zero DCT coefficients generated per chunk. These average values (11.42, 18.84, 55.58, and 99.65) are then designated as the four target complexities for Stable-FAST, corresponding to level 1 through level 4, respectively. This procedure simultaneously defined the first FAST baseline, FAST-C, where each variant (e.g., FAST-C at level 2) uses the fixed chunk length (e.g., 10) whose resulting average complexity matches the corresponding target complexity of Stable-FAST.

2. Deriving average chunk lengths of Stable-FAST and defining FAST-L. After training Stable-FAST tokenizers with the four target complexities, we defined the second FAST baseline. For the chunks encoded by each of the Stable-FAST tokenizers, we calculated the overall mean chunk length (6, 11, 36, and 63) and then use these lengths as the fixed chunk sizes for the FAST-L baseline. Therefore, FAST-L matches the average chunk length of Stable-FAST.

E Statistical Results of Inference Time

To provide a more quantitative analysis of the inference time instability discussed in the introduction, this section presents a statistical comparison that complements the qualitative plot in Figure 2. We evaluated the inference time per action chunk for policies trained with both the baseline FAST tokenizer and our Stable-FAST method. The evaluation was conducted on a representative task from the LIBERO benchmark, using models initialized with SmolVLM.

As illustrated in Figure 9, while the average inference time per chunk is nearly identical for both methods, Stable-FAST demonstrates a dramatic reduction in variance. This result empirically validates our central claim: by standardizing the complexity of the prediction task, Stable-FAST eliminates the large fluctuations in inference latency inherent to the original FAST tokenizer, leading to a more stable and predictable control loop.

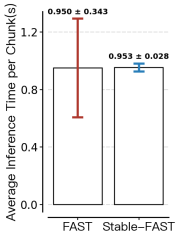

Figure 9: Statistical comparison of inference time per chunk on a representative LIBERO task.

F Taks Progress Definition

To provide a more granular measure of performance on the long-horizon manipulation tasks presented in Table 2, we introduce the Task Progress metric. While the final success rate offers a binary

measure of task completion, it does not capture the degree of partial success in trials that do not ultimately succeed. The Task Progress metric is therefore designed to offer a more nuanced evaluation by quantifying how far the policy progress toward the final goal.

The metric is normalized for each task to represent the percentage of key objectives completed. The specific calculations for the two long-horizon tasks are as follows.

Table bussing. The progress is calculated as the percentage of trash items that are correctly placed into the trash bin relative to the total number of trash items.

Sorting. The progress is defined as the percentage of wooden blocks successfully placed into their designated containers in the correct sequence.

F.1 Limitation and Future Work

While this work presents a logical analysis of inference instability in autoregressive VLAs and offers Stable-FAST as an effective solution, we acknowledge several limitations due to resource and time constraints, which also point toward future works.

First, the scalability of Stable-FAST on massive datasets remains to be explored. Our experiments are conducted on established but relatively constrained benchmarks. It is an open question how the distribution of action complexities will evolve as robot datasets scale up. A more complex and diverse dataset may challenge Stable-FAST for establishing a consistent target complexity. Future work should investigate the robustness and performance of Stable-FAST on large-scale, heterogeneous data.

Second, we do not complete real-world experiments for all VLM backbones, notably Paligemma. Similarly, our experiments are confined to a single robotic arm embodiment. We plan to extend our evaluation to more complex setups, such as bimanual manipulation and other diverse embodiments. Furthermore, we intend to incorporate more tasks that specifically demand dexterous, high-frequency control to provide a more specific comparison against the FAST.

Finally, while we observe a significant reduction in total task execution time, the precise underlying mechanisms for this improvement require a more detailed investigation.

F.2 THE STABLE-FAST ALGORITHM

Algorithm 1 The Stable-FAST data preparation and training pipeline

Require: Raw action trajectory dataset $\mathcal{D}_{\text{traj}}$, initial chunk length H_{initial}, DCT quantization scale γ.
Ensure: Trained Stable-FAST tokenizer $\mathcal{T}_{\text{stable}}$, trained VLA policy π_θ.

PART I: Calibrate target complexity
Initialize complexity list $L_{\text{complexity}} \leftarrow []$
for each trajectory $\tau \in \mathcal{D}_{\text{traj}}$ **do**
  for each valid start time t in τ **do**
    $a_{\text{chunk}} \leftarrow \tau[t : t + H_{\text{initial}}]$
    $C_{\text{dct}} \leftarrow \text{round}(\gamma \cdot \text{DCT}(a_{\text{chunk}}))$ ▷ Compute quantized DCT coefficients
    $C_{\text{current}} \leftarrow \text{count\_non\_zero}(C_{\text{dct}})$
    Append C_{current} to $L_{\text{complexity}}$
  end for
end for
$C_{\text{target}} \leftarrow \text{mean}(L_{\text{complexity}})$ ▷ Establish the average target complexity
PART II: Generate stable-complexity action chunks
Initialize new dataset $\mathcal{D}_{\text{stable}} \leftarrow []$
for each trajectory $\tau \in \mathcal{D}_{\text{traj}}$ **do**
  for each valid start time t in τ **do**
    $H \leftarrow 1$
    $C_{\text{prev}} \leftarrow 0$
    loop
      if $t + H > \text{length}(\tau)$ **then**
        ▷ Handle boundary case by padding with the last action
        $a_{\text{pad}} \leftarrow \tau[\text{length}(\tau) - 1]$
        $a_{\text{chunk}} \leftarrow \text{concatenate}(\tau[t : \text{length}(\tau) - 1], \text{repeat}(a_{\text{pad}}, t + H - \text{length}(\tau)))$
      else
        $a_{\text{chunk}} \leftarrow \tau[t : t + H]$
      end if
      $C_{\text{dct}} \leftarrow \text{round}(\gamma \cdot \text{DCT}(a_{\text{chunk}}))$
      $C_{\text{current}} \leftarrow \text{count\_non\_zero}(C_{\text{dct}})$
      if $C_{\text{current}} \geq C_{\text{target}}$ **then**
        if $\text{abs}(C_{\text{current}} - C_{\text{target}}) < \text{abs}(C_{\text{prev}} - C_{\text{target}})$ **then**
          ▷ Chunk of length H is closer to the target complexity
          Append a_{chunk} to $\mathcal{D}_{\text{stable}}$
        else
          ▷ Chunk of length H-1 was closer, reconstruct and append it
          if $t + H - 1 > \text{length}(\tau)$ **then**
            $a_{\text{pad}} \leftarrow \tau[\text{end}]$
            $a_{\text{chunk\_prev}} \leftarrow \text{concatenate}(\tau[t : \text{end}], \text{repeat}(a_{\text{pad}}, t + H - 1 - \text{length}(\tau)))$
          else
            $a_{\text{chunk\_prev}} \leftarrow \tau[t : t + H - 1]$
          end if
          Append $a_{\text{chunk\_prev}}$ to $\mathcal{D}_{\text{stable}}$
        end if
        break ▷ Found the best chunk for this start time
      end if
      $C_{\text{prev}} \leftarrow C_{\text{current}}$
      $H \leftarrow H + 1$
    end loop
  end for
end for
PART III: Train tokenizer and VLA policy
Train BPE tokenizer $\mathcal{T}_{\text{stable}}$ on quantized DCT coefficients from $\mathcal{D}_{\text{stable}}$.
Tokenize all chunks in $\mathcal{D}_{\text{stable}}$ using $\mathcal{T}_{\text{stable}}$ to get tokenized dataset $\mathcal{D}_{\text{tokenized}}$.
Train VLA policy π_θ on observations paired with token sequences from $\mathcal{D}_{\text{tokenized}}$.