TA-VLA: Elucidating the Design Space of Torque-aware Vision-Language-Action Models

Zongzheng Zhang[*1], **Haobo Xu**[*2], **Zhuo Yang**[*1],
Chenghao Yue[1], **Zehao Lin**[1], **Huan-ang Gao**[1], **Ziwei Wang**[3], **Hao Zhao**[†1,2]

[*]Equal Contribution; [†]Corresponding author

[1] Beijing Academy of Artificial Intelligence, BAAI
[2] Institute for AI Industry Research (AIR), Tsinghua Univeristy
[3] Nanyang Technological University

zhaohao@air.tsinghua.edu.cn

https://zzongzheng0918.github.io/Torque-Aware-VLA.github.io/

Abstract: Many robotic manipulation tasks require sensing and responding to force signals such as torque to assess whether the task has been successfully completed and to enable closed-loop control. However, current Vision-Language-Action (VLA) models lack the ability to integrate such subtle physical feedback. In this work, we explore Torque-aware VLA models, aiming to bridge this gap by systematically studying the design space for incorporating torque signals into existing VLA architectures. We identify and evaluate several strategies, leading to three key findings. First, introducing torque adapters into the decoder consistently outperforms inserting them into the encoder. This is because torque signals align more closely with the decoder's input, and the decoder is more sensitive to variations in input. Second, torque history proves to be a critical signal. We find that the most effective way to incorporate it is by summarizing the entire history into a single token, as this preserves the original input pattern of the decoder. Third, inspired by joint prediction and planning paradigms in autonomous driving, we propose predicting torque as an auxiliary output, which further improves performance. This strategy encourages the model to build a physically grounded internal representation of interaction dynamics. Extensive quantitative and qualitative experiments across contact-rich manipulation benchmarks validate our findings.

Keywords: Torque Integration, VLA Models

1 Introduction

Understanding physical interactions through force cues is essential for mastering real-world robotic manipulation. One particularly informative signal is joint torque, which reflects subtle variations in end-effector contact dynamics without requiring external force sensors [1, 2, 3]. As shown in Figure 1(a), different outcomes in a seemingly simple task like charger insertion—no contact, failed insertion, and successful plug-in—can be clearly distinguished by the joint torque profiles of a 7-DoF arm. These torque responses offer rich physical context that is otherwise imperceptible from RGB observations alone. However, despite the growing success of Vision-Language-Action (VLA) models [4, 5, 6, 7, 8] in bridging vision and control, their ability to interpret and leverage such physical feedback remains limited. Our work aims to bridge this gap by integrating torque signals into pretrained VLA models, enabling contact-sensitive decision-making without compromising generalization or scalability.

The challenge lies in how to embed torque into VLA architectures. Torque is a proprioceptive signal, structurally different from image and language inputs, and varies across time, especially during contact-rich phases. As illustrated in Figure 1(c), multiple torque integration strategies exist across three axes—when (immediate vs. historical vs. predictive), where (encoder vs. decoder), and

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

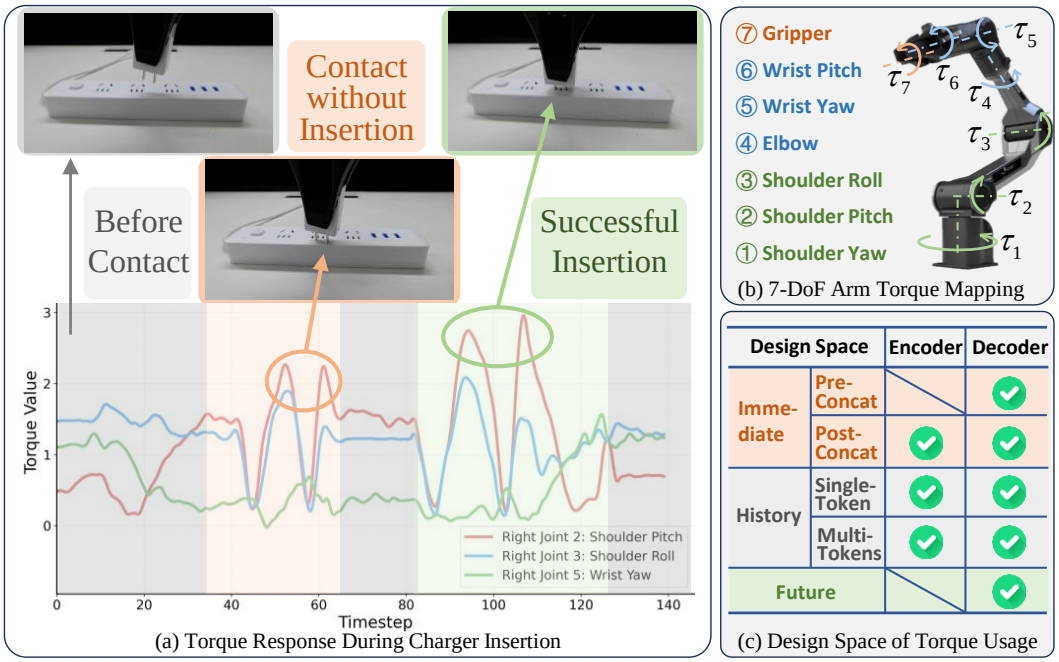


Figure 1: (a) Torque response of the 7-DoF arm during a charger-insertion task. Shaded gray regions mark periods of no contact, where torques remain nearly flat. The orange-tinted segment shows a failed insertion attempt—contact is made but the plug does not enter the socket, producing only small torque fluctuations. The green-tinted segment highlights a successful insertion, characterized by large, distinctive torque spikes as the plug seats fully. (b) Visualization of the 7-DoF robot arm, highlighting joint torque mappings. (c) Design space of torque-based features explored in this work, spanning current, historical, and future signals.

how (single token vs. multi-token). These options form a broad design space, but lack systematic understanding. Our motivation is thus twofold: (i) identify the most effective design choices for torque-aware VLA models, and (ii) derive generalizable principles that guide future integration of physical modalities.

Our first insight is that torque signals should be integrated into the decoder, not the encoder. As validated through HSIC analysis [9] and ablation (Sec. 4.1), decoder-side integration aligns torque with other proprioceptive signals (like joint angles) during action generation. This placement leverages the decoder's higher sensitivity to fine-grained variations—critical in contact-rich scenarios (e.g., distinguishing a failed vs. successful plug in Figure 1(a)).

Our second insight is that historical torque information is more informative than single-frame input. However, injecting multiple tokens can disrupt the decoder's learned input patterns. We find that encoding the entire torque history into a single token in the decoder (Figure 4(c)) balances informativeness with architectural stability. This design choice outperforms both per-frame and encoder-side history integration (Sec. 4.2), enabling robust temporal modeling of contact dynamics, as seen during insertion and retries in Figure 7.

Our third insight is that predicting future torque alongside actions helps build a physically grounded latent space. Inspired by multi-task architectures in autonomous driving [10], we propose a unified action–torque diffusion model (Sec. 5), which allows the policy not only to act but also to anticipate physical consequences (see prediction curves in Figure 6). This auxiliary task encourages the model to internalize contact dynamics beyond observation alone.

Finally, we validate our full system with extensive real-world experiments across 10 diverse tasks—including five contact-rich ones where torque feedback is critical. Our final model (π_0+obs+obj) achieves consistent gains over strong VLA baselines [11, 7, 6] (Table 5), and gen-

eralizes across both model architectures and robot embodiments. These results confirm that torque-aware VLA models not only improve task success but also increase robustness and generalization.

In summary, our contributions are: (1) We propose a systematic design space for torque-aware VLA modeling, spanning where and how torque is integrated (Figure 1(c)). (2) We find that decoder-side, single-token torque history yields the best proprioceptive alignment and performance. (3) We introduce a unified action–torque diffusion model that enables anticipatory learning through torque prediction. (4) We demonstrate significant performance gains on contact-rich manipulation tasks across models and embodiments.

2 Related Work

Vision-Language-Action Model. Recently, Large Language Models (LLMs) [12, 13, 14, 15] and Vision-Language Models (VLMs) [16, 17, 18, 19, 20, 21, 22, 23] have achieved remarkable success, while generative models have enabled continuous outputs such as image generation [24, 25, 26, 27]. These techniques pave the way for the advent of VLA models, which combine visual perception, language understanding, and action generation abilities, and demonstrate strong generalizability, utilizing millions of training data samples including different tasks and devices [28, 29, 30, 31, 32, 33, 34]. Recent works regarding VLA models can be categorized into several modes based on action generation methods, including diffusion policy-based models [5, 7], flow matching-based models [6], and autoregressive generation models [35, 4, 36]. For example, Octo [5] and RDT-1B [7] utilize a diffusion head and a transformer backbone to predict actions, while π_0 [6] uses conditional flow matching to generate high-frequency action sequences. Models with hybrid architectures further combine multiple generative techniques to leverage the strengths of each. For instance, HybridVLA [37] merges diffusion and autoregressive approaches within a single model.

Imitation Learning with Force/Torque. While most of the existing work regarding imitation learning utilizes joint-position and visual information [6, 8], force/torque information as an extra input is gaining more and more attention. Recent studies have demonstrated that force/torque signals could equip controlling policies with the ability to handle a wide range of real-world tasks, which include subtle and precise manipulation [38, 39, 40, 41, 42, 43]. From the perspective of sources of force/torque, most approaches rely on additional sensors to obtain 6D wrench measurements [44, 45, 46, 47, 43], which leads to higher economic costs and limitations in harsh operating conditions. Furthermore, although some works are trying to incorporate force/torque information with visual and text inputs, they typically train policies from scratch and fail to leverage the advantages of pretrained VLA models [48, 49, 50, 51]. For example, FACTR [52] requires a complex training pipeline to align different modalities and lacks flexibility. In contrast, incorporating the force modality into pretrained VLA models offers two major benefits. 1) VLA models already possess a strong foundation in cross-modal learning, having been trained on large-scale datasets; thus, integrating an additional modality is easier. 2) VLA models typically learn shared feature representations across modalities, making it more efficient to accommodate new modality. In this work, we systematically explore ways to enrich pretrained VLA models with force information, allowing them to act as world models that accurately perceive and predict the environment through a unified understanding of vision, force, and instruction over historic, immediate, and future states.

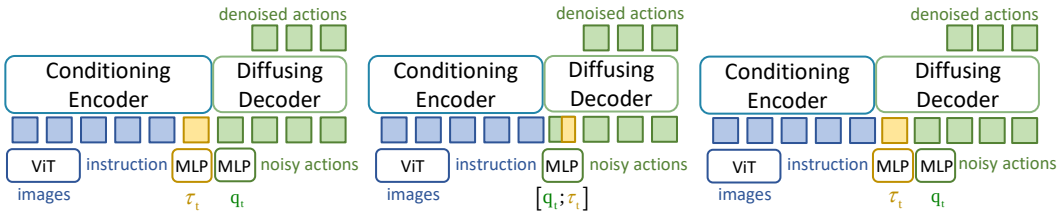
3 Torques are Good Indicators for End-effector Status

In robotic manipulation, external contact at the end-effector induces mechanical responses across the entire kinematic chain. These responses manifest as observable variations in joint torques. In this section, we formalize how joint torque signals encode contact force information via the mechanical arm's differential kinematics and dynamics.

Formulation. Suppose that the manipulator has n degrees of freedom, with joint configuration vector $\boldsymbol{q} \in \mathbb{R}^n$. The full-body dynamics in the presence of external contact are given by:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau}_{\text{cmd}} + \boldsymbol{\tau}_{\text{ext}}, \tag{1}$$

where $\boldsymbol{\tau}_{\text{cmd}} \in \mathbb{R}^n$ is the commanded torque, and $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^n$ is the torque contribution caused by external forces applied at the end-effector. The term $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in$



(a) Enc Architecture (b) DePre Architecture (c) DePost Architecture

Figure 2: Architectures for embedding torque signals.

$\mathbb{R}^{n\times n}$ is the Coriolis and centrifugal force matrix, and $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^n$ is the gravity torque vector. Here, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint angular velocity vector, and $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint angular acceleration vector.

Mapping. Assume that the end-effector makes contact with a rigid environment and experiences a spatial force (wrench) denoted as $\boldsymbol{F}_{\text{ext}} \in \mathbb{R}^6$. Given that the virtual end-effector displacement is related to joint displacement by $\delta\boldsymbol{x} = \boldsymbol{J}(\boldsymbol{q})\delta\boldsymbol{q}$, we obtain:

$$\boldsymbol{\tau}_{\text{ext}} = \boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{F}_{\text{ext}} \in \mathbb{R}^n. \tag{2}$$

Here, $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix, which maps the velocity from the end-effector space to the joint space. This equation is fundamental: it states that any external force acting on the end-effector is projected back into joint space through the transpose of the Jacobian matrix. Therefore, when a contact event occurs (e.g., the robot touches a surface), the resulting torque signal can be decomposed as:

$$\boldsymbol{\tau}_{\text{measured}} = \boldsymbol{\tau}_{\text{model}} + \boldsymbol{J}^\top(\boldsymbol{q})\boldsymbol{F}_{\text{ext}}. \tag{3}$$

Here, $\boldsymbol{\tau}_{\text{measured}}$ is the observed joint torque, and $\boldsymbol{\tau}_{\text{model}}$ accounts for the expected torque due to internal dynamics. This expression shows that observing variations in joint torques allows us to infer the net external wrench acting on the end-effector, provided that the manipulator dynamics are accurately modeled.

Conclusion. The joint torque vector $\boldsymbol{\tau}_{\text{measured}}$ inherently carries information about external contacts through the relationship Eq. (2). This result forms the theoretical basis for torque-based contact estimation, where joint torque deviations from the nominal model are used to infer spatial interaction forces, enabling sensorless force estimation, collision detection, and compliant manipulation.

4 Sense What Was: Torques as Observations

In this section, we integrate torque signals as an additional observation into the VLA framework and investigate its effects. Most VLA models consist of two main components: a *conditioning encoder* and a *denoising decoder*, which we refer to simply as the *encoder* and the *decoder*. The encoder percepts the environment, while the decoder yields actions. To be specific, the encoder processes images inputs I and language instruction L into a unified latent space to construct contextual representations, and the decoder then progressively refines noisy inputs $\hat{\boldsymbol{A}}_{t:t+H}$ to generate action sequences $\boldsymbol{A}_{t:t+H}$. For example, RDT [7] applies cross-attention over visual and language features to form the conditioning and employs a denoising backbone operating on low-dimensional proprioceptive inputs and noised action chunks. Similarly, π_0 [6] leverages a PaliGemma [53] backbone to fuse vision and language inputs, followed by an action decoder.

Using π_0 as a representative case, we explore the design space of torques as inputs with following questions: (1) where to incorporate torque signals—into conditioning encoder or denoising decoder (Sec. 4.1) and (2) how historical torque signals can be leveraged (Sec. 4.2).

4.1 Where to Embed? Conditioning Encoder vs. Denoising Decoder

At each time step t, the policy π_0 observes multiple RGB images, a textual instruction, and the robot's joint angle state, denoted as $\boldsymbol{o}_t = \left[\boldsymbol{I}_{t_1}, \ldots, \boldsymbol{I}_{t_n}, \boldsymbol{L}_t, \boldsymbol{q}_t\right]$, where $\boldsymbol{I}_{t_i}$ is the feature vector of the i-th image, $\boldsymbol{L}_t$ is the sequence of language tokens, and $\boldsymbol{q}_t$ is the current robot state vector. In the original π_0 architecture, the image features $\{\boldsymbol{I}_{t_i}\}$ together with $\boldsymbol{L}_t$ form the conditioning context, while $\boldsymbol{q}_t$ is provided as a token to the denoising module.

Regarding using torque signals and integrating it into the VLA architecture, we explore two integration ways: integrating τ_t into the encoder's inputs to leverage its multi-modality capabilities, or incorporating τ_t into the decoder alongside $\boldsymbol{q}_t$ to enrich the state representation. Specifically, we evaluate three possible strategies for embedding τ_t (see Figure 2):

- **Encoder Embedding (Enc):** encode τ_t via an adapter into a token which is concatenated with $\{\boldsymbol{I}_{t_1}, \ldots, \boldsymbol{I}_{t_n}, \boldsymbol{L}_t\}$ as an extra conditioning input (Figure 2(a));
- **Decoder Pre-Concatenation Embedding (DePre):** directly integrate τ_t into the zero-padded dimensions of $\boldsymbol{q}_t$, concatenating them to form a single combined token (Figure 2(b));
- **Decoder Post-Concatenation Embedding (DePost):** encode τ_t through an adapter and prepend the resulting token to the action expert's state inputs (Figure 2(c)).

Specifically, we employ an MLP as the torque adapter. We conducted real-world experiments on two contact-rich tasks using the three different architectures. The results are shown in Table 1, which shows that embedding torque signals into decoder outperforms into encoder, and embedding it to a single token outperforms integrating it to the original proprioceptive state token. The reasons for the result can be summarized as follows.

Better Input Alignment. Integrating torque signals τ_t into decoder outperforms placing it in encoder. Since τ_t and joint angles q_t are both proprioceptive signals, fusing them during denoising better exploits their correlation, such as consistency and redundancy in contact-rich interactions. To verify this, we conduct experiments to evaluate the Normalized Hilbert-Schmidt Independence Criterion (HSIC) [9] values between the high-dimensional features of the inputs (together with the action) to evaluate their similarities. Figure 3 shows that torque information is significantly more aligned with joint angle signals. Therefore, torque signals should be integrated in decoder to better enhance the proprioceptive perception.

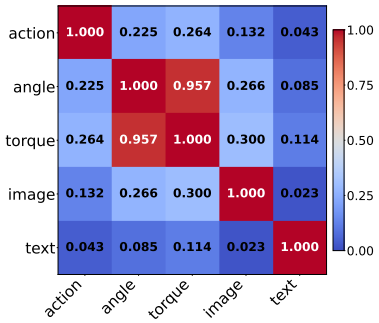


Figure 3: Normalized HSIC values across hidden states from different modality input tokens.

Sensitivity of Decoder. The encoder, designed for diverse, ambiguous vision-language inputs, processes coarser features, whereas decoder is designed to capture subtle variations in inputs. To verify this, we add random noise to each input token of encoder and decoder, respectively, and evaluate the performance. Table 2 shows that, with the effect of noise, decoder shows worse performances, indicating that decoder is more sensitive to the variations in inputs; therefore, introducing τ_t to denoising enables finer utilization of subtle torque variations. Moreover, the Pre-Concatenation method significantly alters the original input token, acting as additional noise, leading to worse performance compared to the Post-Concatenation approach.

Task	π_0	Enc	DePre	DePost
Button Pushing	5/20	7/20	8/20	**10/20**
Charger Plugging	0/20	8/20	11/20	**12/20**

Table 1: Results of different architectures for embedding torque signals.

Task	π_0	Enc-Noised	Dec-Noised
Bottle Pick and Place	14/20	12/20	8/20
Button Pushing	5/20	4/20	0/20

Table 2: Results of noised encoder and decoder with random noise.

4.2 Torque Histories Beat Single Frames

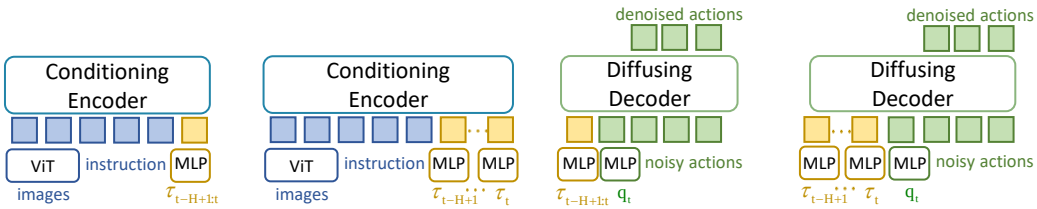
Unlike the fixed language instruction and relatively stable visual observations—which exhibit minimal changes after end-effector contact due to occlusions—torque signals vary significantly upon contact, as illustrated in Figure 1. To capture the dynamic patterns of torques, a single-frame torque input is insufficient. Encoding the history of torque signals provides the VLA model with richer patterns of physical interaction, thereby enabling better performance on contact-rich tasks.

Task	π_0	Enc-1	Enc-H	Dec-1	Dec-H
Button Pushing	5/20	1/20	4/20	**15/20**	9/20
Charger Plugging	0/20	3/20	6/20	**16/20**	7/20

Table 3: Results of different architectures for embedding torque history.

Task	π_0	Enc-Disrupted	Dec-Disrupted
Bottle Pick and Place	14/20	13/20	8/20
Button Pushing	5/20	5/20	2/20

Table 4: Results of disrupted encoder and decoder with extra noised tokens.



(a) Conditioning – 1 Token (b) Conditioning – H Tokens (c) Denoising – 1 Token (d) Denoising – H Tokens

Figure 4: Architectures for embedding torque history.

To investigate the optimal way of encoding torque history, we explore two strategies: (1) frame-wise tokenization, encoding each torque frame $\{\boldsymbol{\tau}_{t-H+1}, \cdots, \boldsymbol{\tau}_t\}$ as a separate token, and (2) aggregate tokenization, encoding the entire history $\boldsymbol{\tau}_{t-H+1:t}$ into a single token. For completeness, we also examine whether historical torque signals should be inserted into the encoder (Figure 4(a)-(b)) or the decoder (Figure 4(c)-(d)). The results are shown in Table 3, indicating that encoding the entire torque history as one single token into the decoder is the best choice. The reason is as follows.

Input Pattern Completeness. Aggregating tokenization outperforms frame-wise tokenization, as the large number of history tokens disrupts the decoder's original input pattern completeness. To verify this hypothesis, we add extra noise tokens to the encoder and decoder, respectively. As shown in Table 4, the added noise tokens easily disrupt the perception ability of the decoder. This also holds when adding extra torque history tokens, which may interfere with the patterns the decoder learned during pretraining. Therefore, even if fewer history tokens may lead to information loss, the effect of disrupting the decoder's state patterns dominates the trade-off. Moreover, in Table 4, the encoder shows robustness to changes in input patterns and performs better with multiple history tokens which contain more information. However, as mentioned in Sec. 4.1, encoding torque signals provides advantages in proprioceptive alignment and finer-grained perception; therefore, a single token of history information to the decoder outperforms other methods.

5 Predict What will Be: Torques as Objectives

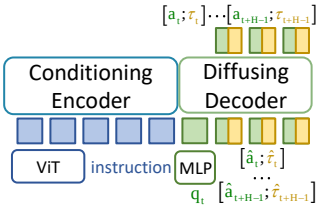


Figure 5: Architectures for Action-Torque Diffusion.

Motivation. Current VLA policies treat modalities purely as observations, missing the opportunity to internalise the robot's own interaction dynamics. Inspired by multi-task planning in autonomous driving [10] and by our findings that torque information is a strong proprioceptive cue in Sec. 4, we propose to **predict future torques together with future actions**. This auxiliary task nudges the model to build a physically grounded latent space, leading to more reliable contact-rich manipulation.

Unified Loss for Action–Torque Diffusion. We now describe in detail how we train the model to predict both actions and torques, maintaining individual losses but sharing diffusion weights for efficiency. Let $\boldsymbol{A}_t \in \mathbb{R}^{H\times d_a}$ denote the action chunk $[a_t, \ldots, a_{t+H-1}]$ and $\boldsymbol{T}_t \in \mathbb{R}^{H\times d_\tau}$ denote the torque chunk $[\tau_t, \ldots, \tau_{t+H-1}]$. The clean joint token can be expressed by $\boldsymbol{Z}_t = [\boldsymbol{A}_t; \boldsymbol{T}_t] \in \mathbb{R}^{H\times(d_a+d_\tau)}$. We sample Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a time step $\alpha \in [0, 1]$, forming a noisy input $\boldsymbol{Z}_t^\alpha = \alpha\, \boldsymbol{Z}_t + (1-\alpha)\, \boldsymbol{\epsilon}$. To ensure both action and torque predictions remain well-calibrated, we define two mean-squared error objectives: $\mathcal{L}_{\text{action}}(\theta) = \mathbb{E}_{\boldsymbol{Z}_t,\boldsymbol{\epsilon},\alpha}\left\|\boldsymbol{v}_\theta(\boldsymbol{Z}_t^\alpha, o_t)_A - (\boldsymbol{\epsilon}_A - \boldsymbol{A}_t)\right\|_2^2$, $\mathcal{L}_{\text{torque}}(\theta) = \mathbb{E}_{\boldsymbol{Z}_t,\boldsymbol{\epsilon},\alpha}\left\|\boldsymbol{v}_\theta(\boldsymbol{Z}_t^\alpha, o_t)_T - (\boldsymbol{\epsilon}_T - \boldsymbol{T}_t)\right\|_2^2$, where $\boldsymbol{v}_\theta(\cdot)_A$ and $\boldsymbol{v}_\theta(\cdot)_T$ denote, respectively, the action- and torque-components of the model's output, and $\boldsymbol{\epsilon}_A$, $\boldsymbol{\epsilon}_T$ are the corresponding slices of the noise. However, unlike the commonly adopted method that multiple types of outputs are predicted through separate modules or shared weights with different projection heads, to save cost and leverage pre-trained weights, we use a single linear layer instead that outputs concatenated action and torque predictions together, then split them back for their respective losses. We adopt the combination of the two losses: $\mathcal{L}_{\text{joint}}(\theta) = \mathcal{L}_{\text{action}}(\theta) + \beta\, \mathcal{L}_{\text{torque}}(\theta)$, where β is a weighting factor balancing action fidelity and torque accuracy.

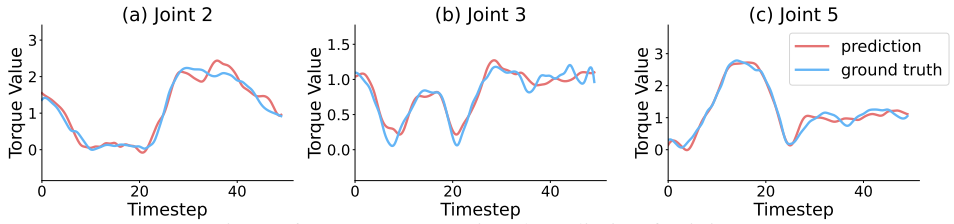


Figure 6: Future torque signal prediction for joints.

Empirical Results. To evaluate the precision of future torques across joints predicted by the action-torque diffusion method, we compare with ground-truth values in validation data. As shown in Figure 6, the predicted torques highly comply with the ground-truth variations, which implies that through the proposed joint diffusion method, the model is able to successfully sense the future changes. This ability will further enable the model to yield better actions (results are shown in Sec. 6.2), because joint torque-action prediction strategy strengthens the model's understanding of contact dynamics by learning the causal relationship between actions and the resulting torque responses. By learning a unified action–torque representation, the model aligns proprioceptive signals with intended motor commands, which enhances performance in contact-rich scenarios.

6 Experiment

6.1 Experimental Setup

Hardware Platform. We use the Cobot Magic ALOHA, a dual-arm robot with 7 degrees of freedom per arm. It is equipped with three D435 depth cameras: two on the wrists and one front-facing. The joint torques are derived from the robot's motors based on the electrical currents supplied to them. Each motor has a specific *current-to-torque constant* k_t, which relates the current i to the generated torque τ, calculated as $\tau = k_t i$. By measuring the current supplied to each motor, we can accurately estimate the joint torque in real-time without the need for external force sensors.

Baselines. We evaluate against strong baselines in robotic manipulation: ACT [11], RDT [7], and π_0 [6]. All models are fine-tuned from publicly available pretrained weights on our collected dataset under the same experimental setup. ACT leverages action chunking with transformers, while RDT and π_0 are two state-of-the-art VLA models with strong cross-task performance.

6.2 Quantitative Results

We evaluate the baseline model together with multiple ways to incorporate torque signals to π_0. Specifically, we adopt (1) *DePost-1 Token Architecture* in Sec. 4 to embed immediate and past torque observations, denoted as π_0**+obs**; (2) the *unified training objective* in Sec.5, denoted as π_0**+obj**; and (3) their combination, denoted as π_0**+obs+obj**. We conduct real-world experiments across 10 tasks—5 contact-rich and 5 regular. Results in Table 5 show that both torque observations and torque-based objectives benefit VLA models. The combined approach leverages the strengths of both, yielding the best overall performance. Also, torque signals improve not only contact-rich tasks but also tasks where torque appears less relevant, indicating their utility across diverse scenarios.

6.3 Visualization

We visualize part of contact-rich and regular tasks the proposed method can achieve. Regarding contact-rich tasks, as shown in Figure 7. When detecting a failed attempt due to abnormal changes in joint torque caused by misalignment or slippage (the second and third images in Figure 7(a)-(b)), leveraging torque feedback, the robot autonomously retries the motion and successfully completes the task on the second attempt (the fourth and fifth images in Figure 7(a)-(b)). Additionally, with torque signals, the robot can complete various regular tasks with high precision (Figure 7(c)).

6.4 Cross Model

To evaluate the generalization capability of torque observations and torque-based objectives across different VLA models, we conduct

Method	Button Pushing	Charger Plugging	Bottle Pick and Place
RDT	4/20	1/20	17/20
RDT + obs + obj	**16/20**	**15/20**	**19/20**

Table 6: **Cross Model Results.** Success rates across contact-rich and regular tasks on RDT.

Task	Contact-rich Task				
Method	Button Pushing	Charger Plugging	USB Plugging	Socket Unplugging	Door Handle Turning
ACT	2/20	0/20	0/20	12/20	0/20
RDT	4/20	1/20	0/20	10/20	0/20
π_0	5/20	0/20	0/20	16/20	2/20
π_0 + obs	15/20	16/20	15/20	**19/20**	13/20
π_0 + obj	11/20	10/20	12/20	**19/20**	12/20
π_0 + obs + obj	**18/20**	**17/20**	**17/20**	**19/20**	**15/20**

Task	Regular Task				
Method	Bottle Pick and Place	Liquid Pouring	Stacking Cubes	Push-to-Position	Opening a Drawer
ACT	15/20	13/20	12/20	13/20	16/20
RDT	17/20	**17/20**	12/20	15/20	16/20
π_0	17/20	16/20	17/20	16/20	**19/20**
π_0 + obs	18/20	16/20	**18/20**	16/20	**19/20**
π_0 + obj	17/20	16/20	17/20	16/20	18/20
π_0 + obs + obj	**19/20**	**17/20**	17/20	**18/20**	18/20

Table 5: **Quantitative Results.** Success rates across 5 contact-rich tasks and 5 regular tasks, each evaluated 20 trials. Our method consistently outperforms baselines, especially in contact-rich tasks.

Figure 7: **Visualization.** (a) Button Pushing: First attempt fails due to misalignment; second succeeds. (b) Door Handle Turning: Initial turn fails; second opens the door. (c) 5 regular tasks.

experiments on RDT [7] across both contact-rich and regular tasks. As shown in Table 6, incorporating both torque observations and the torque-based objective leads to significantly improved performance. These results suggest that the torque integration strategies introduced in Sec.4 and Sec.5 generalize well to other VLA models.

6.5 Cross Embodiment

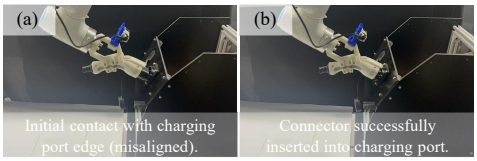


Figure 8: Cross Embodiment Visualization.

To evaluate the generalization capability of our method across different robotic embodiments, we conduct experiments on a ROKAE SR robotic arm. As shown in Figure 8, the robot performs an electric vehicle charger insertion task. After detecting a failed insertion attempt using torque feedback (Figure 8(a)), the robot successfully completes the task on its second attempt (Figure 8(b)).

7 Conclusion

In this paper, we analyze joint torques as effective indicators of end-effector status and explore how to best incorporate them into VLA models. We find that encoding immediate and historical torques as a single decoder token yields the best results. Further, jointly predicting actions and torques with a unified diffusion loss improves performance. Experiments on contact-rich and regular tasks confirm the effectiveness and generalization of both torque-based enhancements.

8 Limitations

The method relies on accurate torque estimation from internal motor currents. This estimation can be affected by motor calibration, sensor noise, or thermal drift, potentially degrading performance in prolonged or high-load tasks. Moreover, while torque signals prove valuable, it remains unclear how scalable the framework is when extending to other physical modalities like tactile sensing or temperature, especially under shared token budgets in transformer architectures. Future work is needed to evaluate robustness in more diverse, real-world scenarios, as well as to explore the alignment and integration of richer multimodal signals.

References

[1] N. Likar and L. Žlajpah. External joint torque-based estimation of contact information. *International Journal of Advanced Robotic Systems*, 11(7):107, 2014.

[2] S. Shan and Q.-C. Pham. Fine robotic manipulation without force/torque sensor. *IEEE Robotics and Automation Letters*, 9(2):1206–1213, 2023.

[3] X. Xu, L. Cheng, L. Miao, X. Zhou, J. Li, and Y. Ke. End-effector contact force estimation for the industrial robot in automated fiber placement processes with dynamic end-load variations. *CIRP Journal of Manufacturing Science and Technology*, 55:390–402, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[5] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[6] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[7] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *International Conference on Learning Representations*, 2025.

[8] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[9] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[10] S. Shi, L. Jiang, D. Dai, and B. Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35: 6531–6543, 2022.

[11] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[13] X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

[14] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[15] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[16] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[17] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024.

[18] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[19] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[20] X. Liu, B. Tian, Z. Wang, R. Wang, K. Sheng, B. Zhang, H. Zhao, and G. Zhou. Delving into shape-aware zero-shot semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2999–3009, 2023.

[21] P. Li, B. Tian, Y. Shi, X. Chen, H. Zhao, G. Zhou, and Y.-Q. Zhang. Toist: Task oriented instance segmentation transformer with noun-pronoun distillation. *Advances in Neural Information Processing Systems*, 35:17597–17611, 2022.

[22] K. Ding, B. Chen, Y. Su, H.-a. Gao, B. Jin, C. Sima, W. Zhang, X. Li, P. Barsch, H. Li, et al. Hint-ad: Holistically aligned interpretability in end-to-end autonomous driving. *arXiv preprint arXiv:2409.06702*, 2024.

[23] B. Jin, Y. Zheng, P. Li, W. Li, Y. Zheng, S. Hu, X. Liu, J. Zhu, Z. Yan, H. Sun, et al. Tod3cap: Towards 3d dense captioning in outdoor scenes. In *European Conference on Computer Vision*, pages 367–384. Springer, 2024.

[24] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[25] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[26] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[27] Z. Zhang, X. Li, S. Zou, G. Chi, S. Li, X. Qiu, G. Wang, G. Zheng, L. Wang, H. Zhao, et al. Chameleon: Fast-slow neuro-symbolic lane topology extraction. *arXiv preprint arXiv:2503.07485*, 2025.

[28] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[29] J. Huang, S. Yong, X. Ma, X. Linghu, P. Li, Y. Wang, Q. Li, S.-C. Zhu, B. Jia, and S. Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023.

[30] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[31] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[32] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[33] A. Jiang, Y. Gao, Z. Sun, Y. Wang, J. Wang, J. Chai, Q. Cao, Y. Heng, H. Jiang, Y. Dong, et al. Diffvla: Vision-language guided diffusion planning for autonomous driving. *arXiv preprint arXiv:2505.19381*, 2025.

[34] H. Chi, H.-a. Gao, Z. Liu, J. Liu, C. Liu, J. Li, K. Yang, Y. Yu, Z. Wang, W. Li, et al. Impromptu vla: Open weights and open data for driving vision-language-action models. *arXiv preprint arXiv:2505.23757*, 2025.

[35] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[36] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[37] J. Liu, H. Chen, P. An, Z. Liu, R. Zhang, C. Gu, X. Li, Z. Guo, S. Chen, M. Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*, 2025.

[38] Y. Chen, A. Sipos, M. Van der Merwe, and N. Fazeli. Visuo-tactile transformers for manipulation. *arXiv preprint arXiv:2210.00121*, 2022.

[39] W. Liu, J. Wang, Y. Wang, W. Wang, and C. Lu. Forcemimic: Force-centric imitation learning with force-motion capture system for contact-rich manipulation. *arXiv preprint arXiv:2410.07554*, 2024.

[40] K. Ding, B. Chen, R. Wu, Y. Li, Z. Zhang, H.-a. Gao, S. Li, G. Zhou, Y. Zhu, H. Dong, et al. Preafford: Universal affordance-based pre-grasping for diverse objects and environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7278–7285. IEEE, 2024.

[41] Y. Hou, Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, S. Feng, B. Burchfiel, and S. Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control. *arXiv preprint arXiv:2410.09309*, 2024.

[42] C. Chen, Z. Yu, H. Choi, M. Cutkosky, and J. Bohg. Dexforce: Extracting force-informed actions from kinesthetic demonstrations for dexterous manipulation. *arXiv preprint arXiv:2501.10356*, 2025.

[43] W. van den Bogert, M. Iyengar, and N. Fazeli. Built different: Tactile perception to overcome cross-embodiment capability differences in collaborative manipulation. *arXiv e-prints*, pages arXiv–2409, 2024.

[44] Z. He, H. Fang, J. Chen, H.-S. Fang, and C. Lu. Foar: Force-aware reactive policy for contact-rich robotic manipulation. *arXiv preprint arXiv:2411.15753*, 2024.

[45] M. Aburub, C. C. Beltran-Hernandez, T. Kamijo, and M. Hamaya. Learning diffusion policies from demonstrations for compliant contact-rich manipulation. *arXiv preprint arXiv:2410.19235*, 2024.

[46] B. Huang, Y. Wang, X. Yang, Y. Luo, and Y. Li. 3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing. *arXiv preprint arXiv:2410.24091*, 2024.

[47] T. Kamijo, C. C. Beltran-Hernandez, and M. Hamaya. Learning variable compliance control from a few demonstrations for bimanual robot with haptic feedback teleoperation system. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12663–12670. IEEE, 2024.

[48] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. *arXiv preprint arXiv:2503.02881*, 2025.

[49] Y. Wu, Z. Chen, F. Wu, L. Chen, L. Zhang, Z. Bing, A. Swikir, A. Knoll, and S. Haddadin. Tacdiffusion: Force-domain diffusion policy for precise tactile manipulation. *arXiv preprint arXiv:2409.11047*, 2024.

[50] T. Kobayashi, M. Kobayashi, T. Buamanee, and Y. Uranishi. Bi-lat: Bilateral control-based imitation learning via natural language and action chunking with transformers. *arXiv preprint arXiv:2504.01301*, 2025.

[51] K. Li, S. M. Wagh, N. Sharma, S. Bhadani, W. Chen, C. Liu, and P. Kormushev. Haptic-act: Bridging human intuition with compliant robotic manipulation via immersive vr. *arXiv preprint arXiv:2409.11925*, 2024.

[52] J. J. Liu, Y. Li, K. Shaw, T. Tao, R. Salakhutdinov, and D. Pathak. Factr: Force-attending curriculum training for contact-rich policy learning. *arXiv preprint arXiv:2502.17432*, 2025.

[53] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Elucidating the Design Space of Torque-aware Vision-Language-Action Models

A Appendix

In this appendix, we provide a comprehensive elaboration of the technical, experimental, and implementation details of our study. Sec. A.1 presents additional qualitative visualizations to supplement the main text, highlighting the system's performance in various scenarios. Sec. A.2 meticulously derives the wrench-to-torque mapping for a 7-DOF manipulator, including the full spatial Jacobian, joint-specific partitions, and a quasi-static simplification for efficient torque computation. Sec. A.3 and A.4 detail the experimental protocols for the torque-integration and torque-history encoding strategies, ensuring reproducibility. Sec. A.5 outlines the implementation specifics of the joint action-torque diffusion objective, clarifying how torque prediction is achieved alongside action generation. Sec. A.6 provides further insights into the experimental setup, quantitative results, and cross-model as well as cross-embodiment evaluations, ensuring a robust understanding of our results. Sec. A.7 describes the architectural specifications of the baseline VLA models (π_0 and RDT). Sec. A.8 evaluates the system's efficiency in terms of training and inference, demonstrating that our torque-aware enhancements do not significantly impact computational performance. Sec. A.9 ablates the loss weight β for the joint action–torque diffusion objective on Button Pushing. Sec. A.10 compares torque-history aggregation (MLP, RNN, attention) under identical settings. Finally, Sec. A.11 clarifies torque-signal usage and preprocessing.

A.1 Additional Visualizations

We provide additional visualizations of the model's execution process for the tasks described in the main text in Figures 9 and 10. For the contact-rich tasks, the torque response of the Shoulder Pitch, Shoulder Roll, and Wrist Yaw joints during execution is plotted in the last column, similar to Figure 1. We have marked the torque changes corresponding to failed and successful attempts for each task. See our project page for complete videos.

A.2 Detailed Wrench-to-Torque Mapping for a 7-DOF Manipulator

Below we expand the derivation in Sec.3, make explicit the structure of the Jacobian, and specialize it to a 7-DOF arm in which the first six joints are revolute actuators (shoulder→wrist) and the 7-th joint is the gripper's open/close degree of freedom.

A.2.1 Full Spatial Jacobian

For a serial arm with n joints, the geometric Jacobian $J(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ stacks the linear and angular velocity components:

$$J(\boldsymbol{q}) = \begin{bmatrix} J_v(\boldsymbol{q}) \\ J_\omega(\boldsymbol{q}) \end{bmatrix}, \quad J_v, J_\omega \in \mathbb{R}^{3\times n}. \tag{4}$$

For a revolute joint j with axis $\hat{\boldsymbol{z}}_j$ and origin $\boldsymbol{p}_j$ (in the base frame):

$$\begin{aligned} J_{v[:,j]} &= \hat{\boldsymbol{z}}_j \times (\boldsymbol{p}_e - \boldsymbol{p}_j), \\ J_{\omega[:,j]} &= \hat{\boldsymbol{z}}_j, \end{aligned} \tag{5}$$

where $\boldsymbol{p}_e$ is the end-effector position.

A.2.2 Partition for the 7-DOF Arm

Let the joint order be:

$$[q_1, \ldots, q_6, q_7] = \underbrace{[\text{Sh. Yaw, Sh. Pitch, Sh. Roll, Elb., Wr. Yaw, Wr. Pitch,}}_{\text{arm (1-6)}} \quad \underbrace{\text{Gripper open/close}]}_{7}. \tag{6}$$

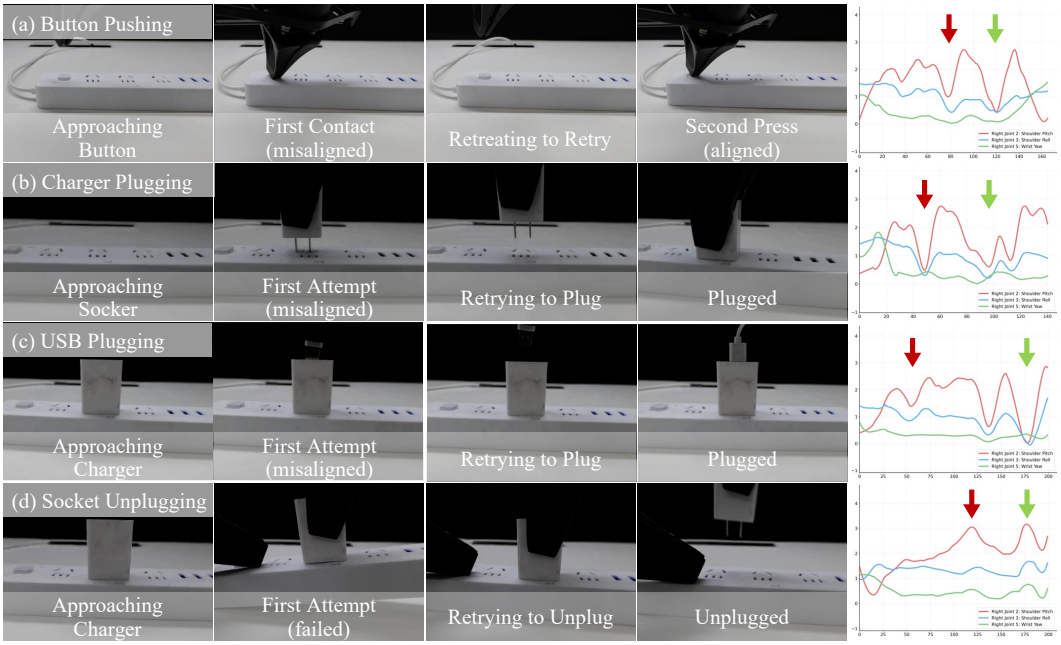


Figure 9: **Visualization of contact-rich tasks.** The last column visualizes the torque response of some joints during the task. For each task, the first failed attempt is marked with a red arrow, and the final successful attempt is marked with a green arrow.

Then:

$$J(\boldsymbol{q}) = [J_{\text{arm}}(\boldsymbol{q}); J_{\text{grip}}(\boldsymbol{q})], \quad J_{\text{arm}} \in \mathbb{R}^{6\times 6}, \quad J_{\text{grip}} \in \mathbb{R}^{6\times 1}. \tag{7}$$

Gripper column. The gripper's opening motion does not change the Cartesian pose of the tool-centre-point (TCP), so:

$$J_{\text{grip}}(\boldsymbol{q}) = \mathbf{0}_{6\times 1} \Rightarrow \tau_{\text{ext},7} = 0. \tag{8}$$

Consequently, forces at the TCP do not back-propagate torque to q_7 :

$$\tau_{\text{ext},7} = J_{\text{grip}}^{\top} \boldsymbol{F}_{\text{ext}} = 0. \tag{9}$$

Arm columns 1:6. Each column is computed using the revolute formula above. For clarity we show the symbolic structure:

$$J_{\text{arm}} = \begin{bmatrix} \hat{z}_1 \times (\boldsymbol{p}_e - \boldsymbol{p}_1) & \cdots & \hat{z}_6 \times (\boldsymbol{p}_e - \boldsymbol{p}_6) \\ \hat{z}_1 & \cdots & \hat{z}_6 \end{bmatrix}. \tag{10}$$

A.2.3 Wrench–Torque Projection

Given the external wrench $\boldsymbol{F}_{\text{ext}} = [\boldsymbol{f}, \boldsymbol{m}]^{\top} \in \mathbb{R}^6$:

$$\boldsymbol{\tau}_{\text{ext}} = J^{\top}(\boldsymbol{q}) \boldsymbol{F}_{\text{ext}} = \begin{bmatrix} J_{\text{arm}}^{\top} \boldsymbol{F}_{\text{ext}} \\ 0 \end{bmatrix} \in \mathbb{R}^7, \tag{11}$$

where $\tau_{\text{ext},1:6} = J_{\text{arm}}^{\top} \boldsymbol{F}_{\text{ext}}$, and $\tau_{\text{ext},7} = 0$.

A.2.4 Quasi-static Simplification

Under low-velocity manipulation $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \approx 0$, Eq. (1) simplifies to:

$$\boldsymbol{\tau}_{\text{measured}} \approx \boldsymbol{G}(\boldsymbol{q}) + \begin{bmatrix} J_{\text{arm}}^{\top} \boldsymbol{F}_{\text{ext}} \\ 0 \end{bmatrix}. \tag{12}$$

Figure 10: **Visualization of general tasks.** (a) Bottle pick and place. (b) Liquid pouring. (c) Stacking cubes. (d) Push-to-position. (e) Opening a drawer.

Subtracting $\boldsymbol{G}(\boldsymbol{q})$ yields the external component $\delta\boldsymbol{\tau}_{\text{ext}}$, where only joints 1–6 are informative for contact detection. Large residuals in those joints directly indicate contact onset, direction, and magnitude, while small residuals in q_7 confirm that gripper actuation alone does not contribute contact-induced torques.

A.2.5 Practical Notes for Implementation

- **Gravity term G:** Estimate via CAD model; recalibrate with payload when grasping heavy objects.
- **Torque from motor currents:** Use manufacturer-provided k_t with thermal compensation.
- **Contact detection:** Threshold $\delta\tau$ values per joint to detect abnormal force events.

This detailed formulation clarifies the exact Jacobian structure, shows why the gripper joint is torque-insensitive to TCP forces, and provides concrete implementation guidance suitable for reproducibility in a top-tier robotics venue.

A.3 Experimental Protocols for Sec. 4.1: Torque-Integration Architectures

Experiment about Comparison between Architectures. For the **Enc** and **DePost** architectures (Figure 2(a)(c)), we randomly initialize a MLP to project the effort token into the latent space. This MLP is structured with layers mapping from an input dimension of 14 (effort dim) to $2 \times width$, followed by a Swish activation, and then mapping from $2 \times width$ to $width$. The $width$ here corresponds to the model's internal dimension: 2048 for the conditioning encoder in the **Enc** architecture and 1024 for the diffusion decoder in the **DePost** architecture. The state input of π_0 is composed of the 14-dimensional joint positions, followed by 18 dimensions of zero-padding. For the **DePre** architecture (Figure 2(b)), we place the 14-dimensional joint efforts into the last 14 positions of this 32-dimensional state.

Experiment about Better Input Alignment. HSIC is a powerful nonparametric measure capable of detecting complex nonlinear relationships between variables without requiring assumptions about their distribution. Normalized HSIC provides a value between 0 and 1, where higher values indicate

stronger statistical dependence. To evaluate modality alignment, we analyze the π_0 model trained using the **DePost** method on the Button Pushing task. We process input tokens obtained from frames randomly sampled from the training dataset of this task through the model's transformer backbone (18 layers total). We then extract the intermediate representations, specifically the hidden states at the output of the 12th layer, corresponding to these input tokens. For the HSIC computation, the number of these extracted intermediate token representations used for each modality (action, angle, torque, image, and text instruction) is downsampled to 32. The normalized HSIC is then computed pairwise between each combination of modalities using RBF kernels, shown in Figure 3.

Experiment about Sensitivity of Decoder. To assess the sensitivity of the encoder and decoder to input variations, we add additive Gaussian noise with a standard deviation of 0.1 to the input tokens. For the encoder, noise is applied to all input tokens, whereas for the decoder, noise is applied specifically to the state token (the first token in the input sequence). The results are presented in Table 2.

A.4 Experimental Protocols for Sec. 4.2: Torque-History Encoding

Experiment about Comparison between Architectures. For historical effort input, we uniformly sampled 10 frames from the past 2 seconds, including the current frame. In the H Tokens configuration, each of these 14-dimensional effort tokens is processed independently using an MLP with the same architecture as described in Appendix A.3, which projects it into the latent space. Conversely, in the 1 Token configuration, the historical effort from all 10 frames is flattened and concatenated into a single 140-dimensional token before being fed into the MLP.

Experiment about Better Input Pattern Completeness. To investigate the completeness of the input pattern, we introduced an additional token into the input token sequence. This token was sampled from a standard normal distribution and had the same shape as other tokens in the sequence. We then modified the input and autoregressive masks accordingly, following the masking patterns applied to the effort tokens to integrate this new token into the sequence processing. Table 4 shows the results of the experiment.

A.5 Implementation of the Joint Action-Torque Diffusion Objective (Sec. 5)

To enable the model to simultaneously output future effort (torque) for supervision alongside action predictions, we expanded the dimension of the action input and output projection linear layer of the original model (Figure 5). When loading the pre-trained weights, we initialized the portion of the modified weight matrix corresponding to the original action output dimensions with the pre-trained values. The remaining weights, which correspond to the newly added dimensions for future effort prediction, were initialized using smaller values. This initialization strategy is designed to minimally affect the original pre-trained behavior initially, allowing the model to gradually learn the new prediction task during the finetuning process. The predicted future effort sequence has a length $H = 50$ steps, matching the action chunk length. Figure 6 shows an example inference from the model trained on the Button Pushing task. The figure plots the model's predicted future effort per frame against the corresponding ground truth for three selected axes, using an observation frame sampled from the task's validation data as input.

A.6 Additional Details in Sec. 6

A.6.1 Details in Experimental Setup

In the π_0 experiments, we followed the original setup using images from three viewpoints as input: top, left wrist, and right wrist. All π_0 experiments were based on its publicly available pre-trained checkpoint and finetuned both encoder and decoder using LoRA. Training was performed for 30k gradient steps on 4 × NVIDIA L20 GPUs. The RDT experiments used the same GPU setup for full parameter training for 40k gradient steps. For ACT, we use 600 training epochs with a chunk size of 32. All baseline models use AdamW optimizer. Inference was performed using an onboard

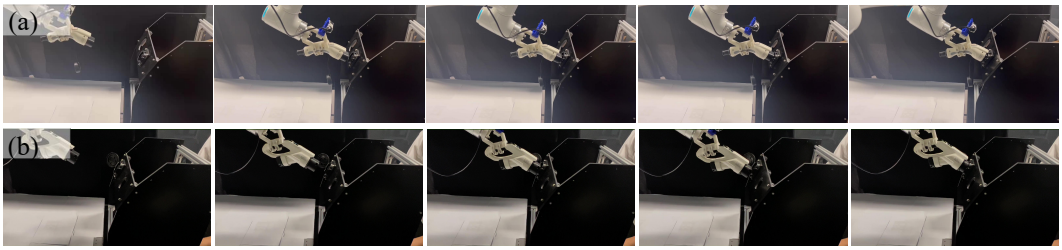


Figure 12: **Cross-Embodiment Task Execution: Charging Connector Insertion.** (a) The robotic manipulator successfully inserts a fast-charging connector into the charging port. (b) The robotic manipulator inserts a slow-charging connector into the charging port.

RTX 4090 GPU. All variants of π_0 used an inference action horizon of 50 steps, and RDT used 64 steps. Other settings remained consistent with the original implementation. For all tasks, 400 demonstrations were collected using teleoperation.

A.6.2 Details about Quantitative Results

In the experiments (Table 5), for $+obj$ settings, we set the value of β to 1. For $+obs + obj$ settings, we set the value of β to 0.1. For the ACT and RDT models, their inference action horizons were 8 and 64 steps, respectively.

A.6.3 Details about Cross Model Results

Regarding the RDT+obs+obj model (Figure 11, Table 6), following the model's approach for language and image adapters, we implement the effort projector as a two-layer MLP with a single GELU activation. This MLP maps the effort input into a 2048-dimensional vector, matching the width of RDT's transformer backbone. The projected effort token is then concatenated after the state token. This combined sequence is further concatenated with the noisy action token, forming the input for the denoising process in RDT. We extended the state space input and output projectors and loaded pre-trained weights in a manner similar to that used in π_0.

A.6.4 Details about Cross Embodiment Results

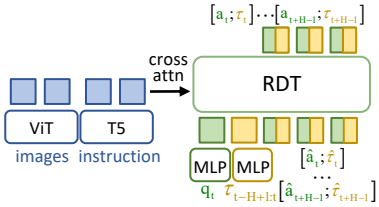


Figure 11: Architecture of RDT+obs+obj model.

To further assess the generalization capability of our torque-aware VLA model across different robotic embodiments, we conducted cross-embodiment experiments using the ROKAE SR robotic arm. Specifically, we trained the π_0+obs+obj with 200 demonstrations of the robot inserting a fast-charging connector into the charging port, using torque feedback to guide the insertion process. The model was trained for 50K gradient steps, and as shown in Figure 12(a), it successfully achieved the fast-charging insertion task with high reliability. To further evaluate generalization, we altered the task setup by replacing the fast-charging port with a slow-charging port. Without any architecture modification, the model was able to adapt seamlessly, successfully inserting the slow-charging connector into the port, as demonstrated in Figure 12(b). This result highlights the model's capacity to generalize torque-aware manipulation strategies to new end-effector configurations, demonstrating robust cross-embodiment performance.

A.7 Architectural Specifications of Baseline VLA Models (π_0 and RDT)

π_0 is a VLA model built by adding a 300M action expert on top of the PaliGemma 3B pretrained VLM backbone and trained on a large-scale cross-embodiment robot dataset. The action expert and the VLM are two independent sets of weights within a single transformer, interacting only through the self-attention layers. During each inference step, the model receives three RGB images,

a language instruction, and the robot's proprioceptive state. The images and language instruction are fed directly into the VLM as conditions. Proprioception is concatenated with a noisy action chunk of length 50 and input to the action expert. π_0 uses Conditional Flow Matching to model the continuous action distribution. The action expert outputs a vector field, and the final action chunk is generated by performing 10 integration steps on this vector field.

RDT (Robotics Diffusion Transformer) is a 1B-parameter Diffusion Transformers (DiTs) model pre-trained using 1M multi-robot trajectories (including 46 datasets). It utilizes a physically interpretable 128-dimensional unified observation and action space, encoding low-dimensional inputs using MLPs with Fourier features. During each inference step, the model's inputs include proprioception, a noisy action chunk (size 64), and the diffusion timestep. Language instruction and observations (including 2 history frames of three images, proprioceptive state, and control frequency) are input to the model as conditions. Images are processed by SigLIP, while language is processed by T5-XXL, this conditional information is alternately injected into different layers of the DiTs using cross attention. The model predicts the denoised action chunk, and the final action is produced through 5 denoising steps.

A.8 System Efficiency

We use the Button Pushing task as an example to compare the training and inference times for π_0, π_0+obs, π_0+obj, and π_0+obs+obj. The training speed is reported as the average throughout the training process measured on 4 × NVIDIA L20 GPUs, and the inference speed is averaged over 10 runs on an RTX 4090 GPU. The results are shown in Table 7 and Table 8, which indicate that the proposed designs do not significantly affect efficiency.

Model Variant	Training Time (s/iter)
π_0	1.703
π_0 + obs	1.628
π_0 + obj	1.648
π_0 + obs + obj	1.640

Table 7: Training Time for Different Designs

Model Variant	Inference Time (ms)
π_0	90.70
π_0 + obs	90.81
π_0 + obj	90.61
π_0 + obs + obj	93.96

Table 8: Inference Time for Different Designs

A.9 Ablation Studies for Hyperparameter β

We test β on π_0+obj and π_0+obs+obj in *button pushing* task (Tab. 9). For π_0+obj: Success rises from 6/20 at $\beta = 0.01$ and reaches a plateau from 0.2 to 1, so we let $\beta = 1$. For π_0+obs+obj: Peak success (18/20) occurs at lower β before dropping at higher values, so we set it as 0.1 to balance the two newly introduced components: auxiliary loss and torque observation modality.

β	0.01	0.1	0.2	0.5	1
π_0 + obj	6/20	8/20	10/20	9/20	**11**/20
π_0 + obs + obj	14/20	**18**/20	**18**/20	15/20	12/20

Table 9: Ablation study on β.

Aggregation	MLP	RNN	Attention
π_0 + obs	**15**/20	7/20	13/20
π_0 + obs + obj	**18**/20	10/20	17/20

Table 10: Inference Time for Different Designs

A.10 Ablation Studies for Torque Aggregation Methods

We compare three aggregation methods for torque history in *button pushing* task, using the same hyperparameters. As shown in Table 10, a simple MLP outperforms others. We attribute it to the parameter efficiency: with limited fine-tuning data, more complex sequence models (RNN/attention) tend to underfit.

A.11 Torque Signal Interpretation and Preprocessing

We use torque signals from **all 7 joints**, not only the end-effector. Due to robot dynamics, some joints—especially near the shoulder (e.g., Joint 2 in Figure 1(b)) exhibit variation during non-contact

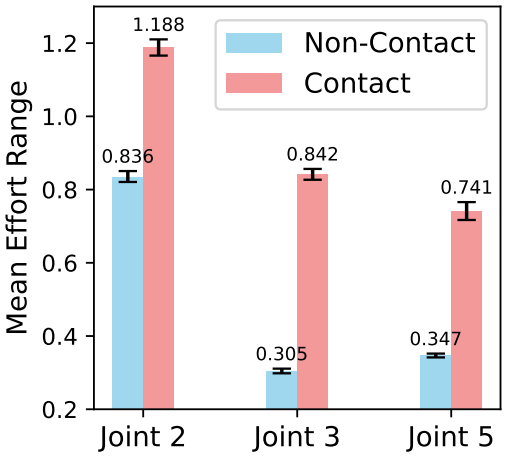


Figure 13: Torque Signals

motion (e.g., 20–36 timestamps in Figure 1(a)). These changes are not noise, and are much smaller than contact-induced signals (Figure 13). Additionally, to ensure stability and keep meaningful patterns, we normalize each joint using training data statistics, as is done for states and actions.