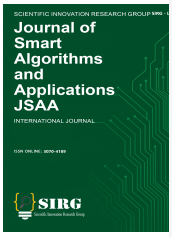

Journal of Smart Algorithms and Applications
JSAA
ISSN: 3070-4189/
Journal Homepage
https://pub.scientificirg.com/index.php/JSAA

Weight-Tied Adaptive Recursive Vision–Language–Action Transformer for Efficient Multimodal Robotic Control

Howaida Allam[a,1] , **Inam Ullah Khan**[b]

[a] *Faculty of Computers, Information and Artificial Intelligence, Lotus University in Minya, Egypt. E-mail: howaidaallam26@gmail.com*

[b] *Faculty of Engineering and Built Science, Lincoln University College, Petaling Jaya 47301, Malaysia; Faculty of Computing and Informatics, Multimedia University, Cyberjaya 63000, Malaysia. E-mail: inamullahkhan05@gmail.com*

ABSTRACT

Vision-Language-Action (VLA) models unify perception, language understanding, and control within a single learning framework, enabling robots to execute manipulation tasks specified through natural language and visual observations. Despite recent progress, many existing VLA systems rely on fixed-depth transformer architectures, resulting in high computational cost and limited adaptability to varying task complexity. We introduce an adaptive recursive VLA architecture that decouples reasoning depth from parameter count through iterative transformer refinement with weight-tied layers. The proposed model processes temporally windowed RGB observations, proprioceptive states, and language instructions using pretrained vision–language encoders and lightweight proprioceptive encoding. Multimodal features are integrated via gated fusion and iteratively refined through recursive transformer iterations, enabling variable-depth reasoning without increasing model size. The refined latent representation conditions structured continuous action prediction, including Cartesian end-effector translation, 6D rotation representation, and gripper actuation. Experimental evaluation on the large-scale DROID robotic manipulation dataset demonstrates substantial improvements over non-recursive baselines. The recursive model achieves a mean squared error (MSE) of 0.020, representing an 82.4% reduction compared to the baseline (MSE: 0.1137). Prediction accuracy reaches 66.82% of actions within 0.10 tolerance and 86.15% within 0.20 tolerance. Position prediction achieves correlations exceeding 0.84-0.96 across all axes, while rotation components show correlations ranging from 0.88 to 0.98. The model maintains computational efficiency with only a 1.5× inference-time overhead while achieving an 82% improvement in accuracy. These results validate recursive reasoning as an effective and computationally efficient mechanism for accurate, adaptable multimodal robotic control.

PAPER INFORMATION

HISTORY

Received: 2 January 2026
Revised: 20 March 2026
Accepted: 19 April 2026
Online: 24 April 2026

MSC

68T07; 68R10; 94A60; 68M15

KEYWORDS

Vision-Language-Action; Robotic manipulation; Recursive transformers; Multimodal fusion; Embodied AI.

[1]Corresponding author at Faculty of Computers, Information and Artificial Intelligence, Lotus University in Minya, Egypt. E-mail: howaidaallam26@gmail.com

1. INTRODUCTION

The field of robotic manipulation has witnessed a transformative shift with the emergence of models that integrate visual perception, natural language comprehension, and action generation into unified frameworks. These Vision Language Action (VLA) systems mark a departure from conventional approaches that treated perception, decision-making, and motor execution as isolated components. Instead of maintaining separate pipelines, modern VLA frameworks process camera images and textual commands jointly to produce executable robot movements a design particularly valuable when robots must interpret open-ended instructions in diverse real-world settings.

The success of large scale multimodal pretraining has opened new possibilities for robotic systems. Models like RT-2 [1] demonstrate that knowledge learned from vast image-text datasets can be adapted to guide robot behaviour, while TinyVLA [2] achieves efficient learning by maintaining frozen visual encoders. Combined with diverse robot datasets such as Open-X-Embodiment [3], these developments point toward general-purpose robotic intelligence.

Despite these advances, current VLA implementations face significant constraints. The predominant reliance on transformer architectures with fixed layer counts creates an inflexible relationship between model capacity and computational requirements: simple tasks and complex scenarios receive identical processing depth, leading to either wasted computation or insufficient reasoning. Furthermore, the substantial memory demands of large transformer networks pose serious obstacles for robots operating under real-time constraints or on resource limited hardware [4, 5, 6].

This research addresses these limitations by developing a recursive processing mechanism inspired by weight sharing across computational iterations. Drawing from Deep Equilibrium Models [7], the proposed approach applies transformer layers iteratively with identical parameters, effectively increasing reasoning depth without expanding model size. This allows the system to adjust its computational investment based on situational demands: more iterations are allocated for ambiguous observations, while straightforward cases maintain efficiency.

The proposed system, named Adaptive Recursive VLA Transformer (ARVLAT), processes sequences of RGB frames from multiple viewpoints alongside robot state measurements and text-based task descriptions. Visual and linguistic information flows through frozen CLIP encoders [8], providing rich semantic representations at manageable cost. Robot proprioceptive data receives separate encoding through a compact neural network. A learnable gating mechanism then combines these multimodal signals, dynamically adjusting their relative influence based on current task requirements and environmental conditions. The integrated representation undergoes iterative refinement through a weight-shared transformer encoder, enabling variable-depth reasoning. Specialized output networks then extract continuous control commands from the refined representation. The system is trained end-to-end on the DROID dataset [9], which contains approximately 76,000 manipulation sequences across 564 environments and 86 task categories.

Empirical evaluation reveals substantial gains: mean squared error (MSE) of 0.020 (an 82% improvement over the baseline), 66.82% of predicted actions falling within a 0.10 tolerance, and correlation coefficients exceeding 0.84 across all spatial control dimensions. Recursive processing achieves these results with only 1.5× inference overhead and approximately 15 million trainable parameters (excluding the frozen CLIP encoders).

This work makes the following contributions:

1. A recursive, weight-shared transformer architecture for VLA control that decouples reasoning depth from parameter count, enabling variable-depth inference without expanding model size.
2. A multimodal framework combining frozen CLIP encoders, a learnable gated fusion mechanism, temporal windowing, and an early-exit option for efficient inference.
3. Experimental validation on DROID that demonstrates consistent gains across all control dimensions, achieving an 82% reduction in MSE compared to single-pass alternatives.

The remainder of the paper is organized as follows. Section 2 reviews related work on vision–language–action models, hierarchical control, and recursive architectures. Section 3 formalizes the VLA problem under temporally windowed multimodal observations. Section 4.2 presents the proposed architecture, including multimodal encoding, gated fusion, temporal reasoning, and recursive refinement. Section 5 presents comprehensive experimental results and quantitative evaluations. Section 6 discusses the

findings, limitations, and practical implications. Finally, Section 7 concludes the paper and discusses future research directions.

2. RELATED WORK

Research in robotic manipulation has increasingly focused on systems that process visual input and natural language simultaneously to generate appropriate actions [9]. Modern implementations adapt large-scale vision-language models to produce robot control commands [9, 10], requiring transformer networks that capture dependencies across time [10]. The RT-2 system treats robot actions as discrete tokens predicted through language modeling objectives [1], while TinyVLA maintains frozen visual encoders to achieve efficient learning [2]. Testing on Open-X-Embodiment demonstrates transfer across robot platforms and task distributions [3].

Recognizing the computational burden of monolithic transformer architectures, researchers have explored hierarchical designs [10, 11] and recursive/equilibrium models [7]. The concept of weight-tied layers for variable-depth processing has been explored in language and vision tasks [7, 12, 13], but its application to continuous robot control remains largely unexplored. Our work bridges this gap by combining weight-tied recursive processing with multimodal fusion for robot manipulation. Multimodal fusion has evolved from simple concatenation toward explicit combination mechanisms that learn context-dependent weights [6, 5].

Recent work on efficient VLA architectures [2, 14], online learning [15, 16], and systematic testing frameworks [4] has further shaped the field. Notably, several 2024–2025 studies have underscored the importance of interpretable and efficient control pipelines in embodied AI [17, 18], reinforcing the need for architectures that balance representational capacity with computational tractability. **Table 1** provides a comprehensive comparison of recent VLA approaches, highlighting architectures, performance metrics, and limitations.

Positioning of ARVLAT: Unlike prior methods that either fix network depth (RT-2, TinyVLA) or require task specific pipelines [10], ARVLAT uniquely contributes: (1) a weight-shared recursive transformer applied to continuous robot control for the first time; (2) a learnable gated multimodal fusion strategy that dynamically adjusts modality emphasis; and (3) an early-exit mechanism that adapts computational cost to task complexity at inference time. These combined contributions distinguish ARVLAT from equilibrium-model works [7] which target NLP tasks, and from hierarchical VLA works that still employ fixed-depth processing.

Table 1: Comparison of Recent Vision–Language–Action Models and Related Approaches

Reference	Model	Architecture	Performance	Key Contribution	Limitations
[1]	RT-2	Vision Language Transformer	62% success	Action tokenization, zero shot	Fixed depth, high compute
[2]	TinyVLA	Compact VLA (frozen encoders)	58% manipulation	Reduced model size	Lower accuracy
[4]	VLATest	Testing framework	N/A	Systematic evaluation	Not a model
[7]	Deep Equilibrium	Equilibrium / weight tied	89% reasoning	Weight-tied iteration	Not robotics-specific
[10]	Lang-Reason VLA	Language conditioned	68% grasp	Language reasoning	Task specific
[19]	Joint VLA Grasp	Unified vision language action	72% clutter	Joint modeling	Limited to grasping
Ours	**ARVLAT**	Adaptive Recursive VLA	**66.82% @ 0.10, MSE: 0.020**	Variable depth, 82% gain	Hyperparameter tuning

3. PROBLEM FORMULATION AND THEORETICAL FRAMEWORK

This work addresses the challenge of learning robot control policies that respond to visual observations and textual commands. The problem is framed as a sequential decision process where actions depend on multiple concurrent input streams. At any given moment t, the robot perceives its environment through an RGB camera producing image $I_t \in \mathbb{R}^{H\times W\times 3}$ and internal sensors measuring joint configurations, end-effector position, and gripper state, collectively denoted as $p_t \in \mathbb{R}^{d_p}$. These measurements constitute the observation $o_t = (I_t, p_t)$. Real implementations often incorporate multiple camera angles, each contributing distinct viewpoints that enrich the robot's perceptual awareness.

A text-based task description $l = (w_1, \ldots, w_m)$ provided at episode initialization remains constant throughout, establishing the semantic context that guides behavior and supports generalization to unseen object-action combinations.

Rather than basing decisions on single observations, the proposed approach conditions actions on recent temporal history to capture dynamic patterns. Specifically, decisions consider a window $O_t = \{o_{t-T+1}, \ldots, o_t\}$ spanning T consecutive timesteps, enabling the model to recognize motion trends and short-term dependencies without maintaining unbounded memory. Training pairs each window $\mathcal{W}_t = \{(I_{t-T+1}, p_{t-T+1}), \ldots, (I_t, p_t)\}$ with the corresponding action a_t that should follow. This design separates the question of how much history to consider from architectural choices about network depth. Sliding the window forward by single timesteps generates numerous training examples from each demonstration trajectory, improving data efficiency.

Robot commands take the form of continuous vectors $a_t \in \mathcal{A} \subset \mathbb{R}^{d_a}$ encoding desired changes in gripper position, orientation, and actuation state. Position adjustments $a_{\text{pos}} \in \mathbb{R}^3$ specify Cartesian movements, while orientations use a 6-dimensional representation $a_{\text{rot}} \in \mathbb{R}^6$ that avoids discontinuities inherent in Euler angles or quaternions. Gripper actuation $a_{\text{grip}} \in \mathbb{R}$ indicates opening or closing. Concatenating these components yields a 10-dimensional action vector: $a_t = [a_{\text{pos}}; a_{\text{rot}}; a_{\text{grip}}] \in \mathbb{R}^{10}$.

The objective is to learn a mapping $\pi_\theta : (O_t, l) \to a_t$ parameterized by weights θ that predicts appropriate actions given observation histories and task descriptions. Training proceeds through supervised regression on demonstrated behavior. A fundamental difficulty lies in the typical dependency between model capacity and computational requirements: deeper networks often generalize better but demand more processing time and memory, creating tensions with real-time control needs.

The capacity-efficiency dilemma is resolved through iterative refinement via weight sharing. The system maintains a latent representation $\boldsymbol{z} \in \mathbb{R}^d$ that is progressively refined by repeatedly applying the same transformation. Given an encoded context $X \in \mathbb{R}^{n \times d}$ containing integrated information from observations and task descriptions, each refinement step applies:

$$\boldsymbol{z}_{i+1} = \mathcal{F}_\theta(\boldsymbol{z}_i, X), \tag{1}$$

implementing update function $\mathcal{F}_\theta$ as:

$$\mathcal{F}_\theta(\boldsymbol{z}, X) = \text{LN}\left(\boldsymbol{z} + \text{Attn}(\boldsymbol{z}, X) + \text{FFN}(\boldsymbol{z} + \text{Attn}(\boldsymbol{z}, X))\right). \tag{2}$$

The attention mechanism $\text{Attn}(\cdot, \cdot)$ allows the latent state to selectively query relevant aspects of the context (Equation 3):

$$\text{Attn}(\boldsymbol{z}, X) = \sum_{j=1}^{n} \alpha_j (X_j W_V), \tag{3}$$

where attention coefficients (Equation 4) are:

$$\alpha_j = \frac{\exp\left((\boldsymbol{z}W_Q)(X_j W_K)^\top / \sqrt{d}\right)}{\sum_{k=1}^{n} \exp\left((\boldsymbol{z}W_Q)(X_k W_K)^\top / \sqrt{d}\right)}, \tag{4}$$

derived from learnable projections $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$, while $\text{FFN}(\cdot)$ denotes a two-layer network with GELU nonlinearity.

Applying this transformation N times produces deeper computation without expanding parameters, similar to unrolling a recurrent network or solving fixed-point Equations [7]. The iteration count can vary at inference, providing flexibility to allocate more computation for challenging scenarios while maintaining efficiency elsewhere.

After N refinement cycles, the evolved representation $\boldsymbol{z}_N$ determines the output action through (Equation 5):

$$\pi_\theta(a_t \mid O_t, l) = f_\theta(\boldsymbol{z}_N), \tag{5}$$

where f_θ encompasses specialized prediction networks detailed in Section 4.2.

This design separates reasoning depth from parameter count, enabling computational adaptation while respecting real-time constraints.

4. METHODOLOGY

This section describes the complete methodological framework, including the dataset and preprocessing pipeline, the proposed model architecture with all components, and the training and optimization procedures.

4.1 DROID Dataset and Preprocessing

The experiments rely on the DROID dataset [9], a substantial collection of robot manipulation demonstrations gathered across diverse real-world settings. **Table 2** summarizes key dataset statistics. Similar large-scale robot learning benchmarks include RLBench and LIBERO [20, 21]. The DROID collection encompasses roughly 76,000 behavior sequences amounting to approximately 350 hours of interaction data. These demonstrations span 564 distinct physical locations—including residential spaces, workplaces, and kitchens—and cover 86 different manipulation objectives involving common objects and actions. Multiple operators contributed data from locations across North America, Asia, and Europe, introducing substantial variation in visual appearance, object arrangements, and interaction styles. This diversity proves valuable for learning policies that generalize beyond narrow task distributions, as the dataset exhibits a long-tailed pattern where many manipulation primitives, object types, and linguistic phrasings appear infrequently—forcing models to extract transferable patterns rather than memorizing specific instances.

Preprocessing converts original demonstration recordings into HDF5 files structured for efficient batch sampling and windowed access. Images undergo decoding and bilinear downsampling to 128×128, maintaining uint8 encoding to reduce storage requirements and enable fast memory-mapped loading. Though DROID provides several camera angles, two external viewpoints are utilized, converting images to float32 only during training batch preparation. Proprioceptive measurements combine Cartesian gripper coordinates with aperture state into 7-dimensional vectors per timestep, simplifying subsequent processing by unifying physical state information. Action transformations convert raw control signals into a standardized continuous format combining 3D Cartesian displacement, 6D rotation encoding, and gripper commands. The rotation parameterization extracts the first two columns from rotation matrices, yielding a continuous representation without singularities that facilitates gradient-based learning. Quality filtering optionally excludes demonstrations marked as incomplete, focusing training on successful execution patterns.

Training samples incorporate temporal context through fixed-duration windows. Each sample aggregates $K = 4$ consecutive observations from demonstration trajectories (Equation 6):

$$\mathcal{W}_t = \{(I_{t-K+1}, p_{t-K+1}), (I_{t-K+2}, p_{t-K+2}), \ldots, (I_t, p_t)\}, \tag{6}$$

pairing image sequences I_t and proprioceptive readings p_t with the subsequent action a_t that follows the window's final observation. This coupling ensures the model learns to predict appropriate responses based on recent history while maintaining causal consistency—actions depend only on past observations. Advancing the window by single timesteps when extracting samples from trajectories generates numerous training instances, improving data utilization and pattern diversity. The resulting collection of window-action pairs $(\mathcal{W}_t, a_t)$ supports end-to-end policy learning that accounts for temporal dynamics. **Figure 1** visualizes this windowing approach.

The dataset exhibits a long-tailed distribution: many manipulation primitives, object types, and linguistic phrasings appear infrequently, forcing models to extract transferable patterns rather than memorize specific instances. To address this imbalance, three strategies are applied: (i) percentile-based action normalization to handle outlier trajectories, (ii) weighted trajectory sampling that up-samples underrepresented task categories during training, and (iii) quality filtering that excludes demonstrations marked as incomplete.

4.2 Proposed Model Architecture

The architecture combines several interconnected components to process visual, linguistic, and proprioceptive inputs while maintaining computational efficiency. The design leverages frozen pretrained networks for vision and language, applies learnable fusion to integrate modalities, employs iterative refinement through weight-shared transformers, and generates continuous control outputs through

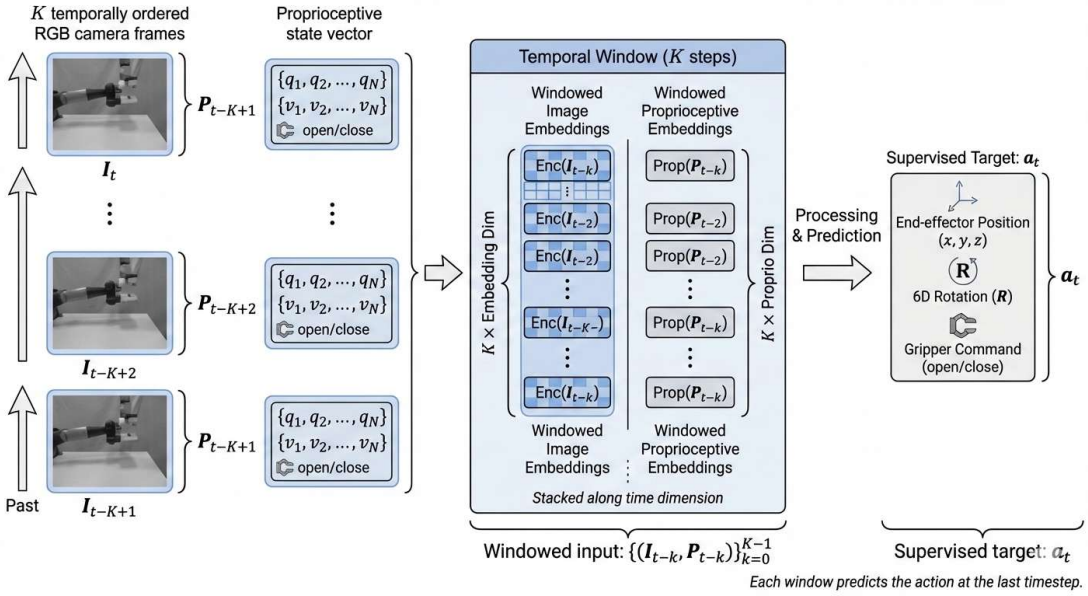

Figure 1: Temporal windowing mechanism showing how consecutive observations are grouped into fixed-length windows paired with target actions at the final timestep.

Table 2: DROID Dataset Statistics and Characteristics

Property	Value	Notes
Total trajectories	∼76,000	∼350 hours of interaction data
Distinct environments	564	Homes, offices, kitchens, labs
Task categories	86	Long-tail distribution; ∼30% tasks have <100 examples
Geographic diversity	North America, Asia, Europe	Multiple operators per region
Action dimensions	10 (3 pos + 6 rot + 1 grip)	6D rotation avoids singularities
Train / Val / Test split	80% / 10% / 10%	Test: 1,500 held-out windows
Class imbalance strategy	Weighted sampling + normalization	Percentile-based action normalization handles outliers

specialized networks. This organization balances model capacity with inference speed, supporting real-time robot operation. **Figure 2** provides an overview of the information flow and component relationships.

Preprocessing converts original demonstration recordings into HDF5 files structured for efficient batch sampling. The pipeline applies the following steps:

- **Image preprocessing:** Frames are decoded from raw recordings and bilinear-downsampled to 128×128 (`uint8`) for compact storage. During training batch preparation, images are converted to `float32`, resized to 224×224, and normalized using CLIP statistics (mean = [0.48145, 0.4578, 0.4082], std = [0.26862, 0.26130, 0.27578]).
- **Proprioceptive encoding:** Cartesian gripper coordinates (3D) plus gripper aperture (1D) are concatenated into 7D vectors per timestep. These are encoded by a compact MLP ($7 \rightarrow 128$ dimensions) with Layer Normalization and ReLU activation.
- **Language embedding:** Task instructions are tokenized via CLIP byte-pair encoding

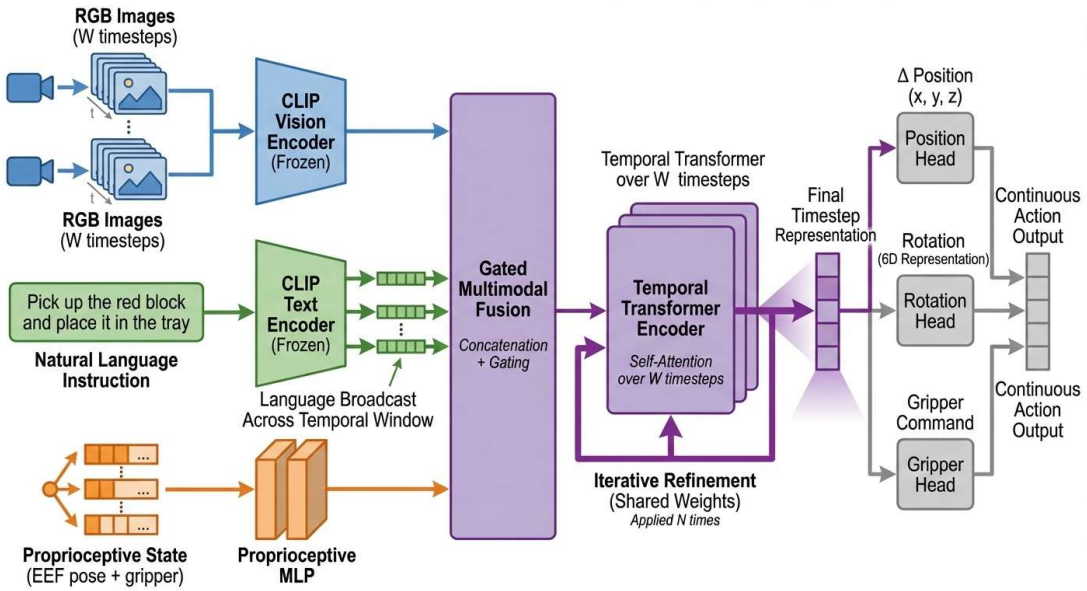

Figure 2: The High-Level Planner performs recursive reasoning over fused multimodal context, conditioning the Low-Level Executor to generate continuous control actions from temporally windowed observations.

(padded/trimmed to 77 tokens), producing 768-dimensional sentence embeddings from the CLIP language transformer. Embeddings are replicated across the temporal window dimension.

- **Action normalization:** Raw control signals are converted to standardized 10D vectors. Rotation is represented as the first two columns of a rotation matrix (6D), avoiding Euler angle singularities. Each action dimension is normalized to $[-1, 1]$ using percentile statistics (1st–99th percentile of training data).

Training samples incorporate temporal context through fixed-duration windows of $K = 4$ consecutive observations:

$$W_t = \{(I_{t-K+1}, p_{t-K+1}), \ldots, (I_t, p_t)\},$$

paired with the subsequent action a_t. Advancing the window by single timesteps when extracting samples from trajectories generates numerous training instances, improving data utilization.

Input processing begins with visual feature extraction through a CLIP Vision Transformer [8], a network pretrained on extensive image-text collections that captures rich semantic patterns. Keeping these pretrained weights frozen during robot learning reduces computational burden while preserving learned visual knowledge. For a batch containing B sequences of W consecutive frames ($\mathbf{v}_{obs} \in \mathbb{R}^{B\times W\times H\times W\times 3}$), images stored compactly as uint8 arrays convert to float32 on the GPU, minimizing memory transfer overhead. After normalizing pixel values according to CLIP's expected statistics and resizing to 224×224 through bilinear sampling, frames pass through the frozen encoder. The output embedding $E_v(\mathbf{v}_{obs}) \in \mathbb{R}^{B\times W\times d_v}$ spans 768 dimensions ($d_v = 768$) for CLIP ViT-B/32. Systems with multiple cameras combine embeddings from different viewpoints through concatenation, enriching the spatial information available for downstream processing.

Text-based task descriptions receive embedding through CLIP's pretrained language component. Task instructions $\mathbf{l} \in \mathcal{L}^B$ undergo tokenization via byte-pair encoding, followed by length normalization through padding or trimming. The transformer's final layer produces a pooled sentence representation $E_\ell(\mathbf{l}) \in \mathbb{R}^{B\times d_\ell}$ of dimensionality $d_\ell = 768$. Because task descriptions persist throughout execution rather than changing per timestep, the system replicates this embedding across the temporal dimension: $\tilde{E}_\ell = \text{repeat}(E_\ell, W) \in \mathbb{R}^{B\times W\times d_\ell}$. Maintaining frozen language encoder weights preserves linguistic knowledge acquired during pretraining while limiting computational demands.

Internal robot measurements $\mathbf{p} \in \mathbb{R}^{B\times W\times 7}$ track gripper position in Cartesian coordinates (three dimensions) plus gripper aperture (one scalar), providing essential feedback for closed-loop control and physical grounding. A compact multilayer perceptron with normalization and nonlinearity encodes these measurements: $E_p(\mathbf{p}) = \mathrm{ReLU}(\mathrm{LN}(W_p\mathbf{p} + b_p))$, yielding representations $E_p(\mathbf{p}) \in \mathbb{R}^{B\times W\times d_p}$ of dimension $d_p = 128$. This relatively modest dimensionality ensures proprioceptive signals contribute substantively to decision-making while avoiding computational bottlenecks or feature dominance during multimodal integration.

Combining vision, language, and proprioception requires careful balancing to ensure each modality contributes appropriately. Recent work has explored various fusion mechanisms including cross-modal attention and adaptive weighting [6, 14]. The fusion strategy employed here uses learnable gating to adjust the relative influence of perceptual and physical state information. The system first merges visual and linguistic embeddings through concatenation and projection into a unified representation space: $F_{vl} = \mathrm{ReLU}(\mathrm{LN}(W_{vl}[E_v; \tilde{E}_\ell])) \in \mathbb{R}^{B\times W\times d_h}$ with hidden dimension $d_h = 512$. A sigmoid-activated network then computes context-dependent weights: $g = \sigma(W_g[F_{vl}; E_p]) \in \mathbb{R}^{B\times W\times d_h}$, which blend the perceptual representation with proprioceptive encoding: $F = g \odot W_v F_{vl} + (1-g) \odot W_p E_p \in \mathbb{R}^{B\times W\times d_h}$. Here $W_v, W_p \in \mathbb{R}^{d_h\times d_h}$ represent trainable projections and $\odot$ denotes pointwise multiplication. The learned coefficients g allow the system to emphasize camera and language inputs when interpreting task semantics while elevating proprioceptive signals when executing precise movements. This flexibility accommodates varying demands across different task phases and environmental conditions.

Maintaining temporal sequence information requires encoding positional relationships. The system adds sinusoidal position codes to fused features: $F' = F + \mathrm{PE}(W) \in \mathbb{R}^{B\times W\times d_h}$, where $\mathrm{PE}(W)$ generates frequency-varying patterns following standard transformer design [11]. A transformer encoder with $L = 6$ layers and $H = 8$ attention heads per layer processes these temporally marked features. The multi-head attention mechanism [11] aggregates information across time through: $\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_h/H}}\right)V$, where Q, K, V represent learned query, key, and value transformations. Each layer combines attention outputs with position-wise feed-forward processing, applying residual connections and normalization. This produces a temporally contextualized representation: $F^{(0)} = \mathrm{TransformerEncoder}(F') \in \mathbb{R}^{B\times W\times d_h}$.

To increase effective reasoning depth without increasing parameter count, the Transformer encoder can be applied recursively with shared weights. Given the initial encoded representation $F^{(0)}$, recursive refinement iterates: $F^{(k+1)} = \mathrm{TransformerEncoder}(F^{(k)})$ for $k = 0, \ldots, N_{\mathrm{rec}} - 1$, where the same Transformer encoder parameters are reused across iterations. This mechanism approximates iterative refinement and recurrent inference while remaining fully differentiable. During training, gradients backpropagate through all unrolled iterations, enabling the model to learn stable update dynamics. At inference time, the number of recursive iterations N_{rec} can be adjusted based on computational constraints or task complexity. The experiments evaluate configurations with $N_{\mathrm{rec}} \in \{1, 2\}$, where $N_{\mathrm{rec}} = 1$ corresponds to a non-recursive baseline. Recursive refinement enables the model to progressively sharpen attention patterns and refine internal representations, which is particularly valuable for resolving perceptual ambiguity, handling occlusions, or performing multi-step spatial reasoning without explicit intermediate supervision.

From the iteratively refined representation, the system extracts features corresponding to the most recent timestep: $h = F^{(N_{\mathrm{rec}})}_{:,-1,:} \in \mathbb{R}^{B\times d_h}$. This vector summarizes the complete multimodal context after N_{rec} refinement cycles. Three distinct output networks convert this summary into control commands spanning different physical degrees of freedom. A position network generates Cartesian translation commands: $\hat{a}_{pos} = \tanh(W_{pos}h + b_{pos}) \in \mathbb{R}^{B\times 3}$, with tanh nonlinearity constraining outputs to $[-1, 1]$. A rotation network produces 6-dimensional orientation vectors: $\hat{a}_{rot} = \tanh(W_{rot}h + b_{rot}) \in \mathbb{R}^{B\times 6}$. This 6D parameterization circumvents singularities and discontinuities present in Euler angle or quaternion representations, facilitating gradient flow during optimization. A gripper network outputs actuation commands: $\hat{a}_{grip} = \tanh(W_{grip}h + b_{grip}) \in \mathbb{R}^{B\times 1}$. Stacking these predictions yields the complete command vector: $\hat{a} = [\hat{a}_{pos}; \hat{a}_{rot}; \hat{a}_{grip}] \in [-1, 1]^{10}$. Deployment requires transforming these normalized predictions back to physical units using stored scaling statistics.

Learning proceeds through supervised regression, minimizing a weighted squared error objective: $\mathcal{L} = \lambda_{pos}\|\hat{a}_{pos} - a_{pos}\|_2^2 + \lambda_{rot}\|\hat{a}_{rot} - a_{rot}\|_2^2 + \lambda_{grip}\|\hat{a}_{grip} - a_{grip}\|_2^2$, where coefficients $\lambda_{pos}, \lambda_{rot}, \lambda_{grip}$ balance the importance of different control dimensions. Tuning these weights accommodates varying error scales and task priorities across position, orientation, and gripper channels. Optimization employs stochastic gradient descent on dataset batches, propagating gradients through all components including

the unrolled recursive iterations. Several design elements promote efficient training and deployment: maintaining frozen pretrained encoders avoids computing gradients for millions of parameters; storing images as uint8 reduces data transfer by 4×; batched tensor operations improve loading throughput; weight-sharing enables deeper processing without parameter expansion; and dedicated output networks support targeted optimization of each control dimension.

4.3 Early Exit Mechanism

To optimize computational efficiency, ARVLAT incorporates an Early Exit Mechanism. The model evaluates the convergence of the hidden states $h_t^{(i)}$ at each iteration i. If the cosine similarity between $h_t^{(i)}$ and $h_t^{(i-1)}$ exceeds a predefined threshold τ (where $\tau \in [0.95, 0.99]$), the recursive process terminates. This ensures that the model allocates more iterations to complex reasoning tasks while exiting early for simpler, repetitive robotic actions.

4.4 Training and Optimization

Stabilizing training across action dimensions requires consistent scaling. Normalization parameters are established by examining sampled episodes, computing 1st and 99th percentile values (p_{01}, p_{99}) for each dimension, then applying linear transformation $a' = 2 \cdot \frac{a - p_{01}}{p_{99} - p_{01}} - 1$ to map actions into $[-1, 1]$. Values exceeding these bounds undergo clipping, though gripper signals may optionally bypass normalization. This percentile-based approach accommodates outliers without allowing extremes to dominate scaling. The resulting normalized targets facilitate stable regression. At deployment, predictions undergo inverse transformation using stored statistics to recover physical units.

Training employs the AdamW optimizer configured with initial learning rate 3×10^{-4} (reduced via cosine schedule), weight decay 1×10^{-4}, per-GPU batch size 32 accumulating over 4 steps (effective batch size 128), 50 maximum epochs with validation-based early stopping, and gradient norm clipping at 1.0 to prevent instability. Loss coefficients ($\lambda_{pos} = 1.0$, $\lambda_{rot} = 1.0$, $\lambda_{grip} = 0.5$) adjust the relative emphasis on position, rotation, and gripper prediction. Position and rotation receive equal weighting to ensure balanced spatial prediction, while the gripper component receives reduced weight (0.5) given its binary nature and lower dimensional complexity. These values emerged from preliminary tuning experiments that systematically varied weight ratios to balance accuracy across control dimensions given their differing magnitudes and behavioral impacts.

Training infrastructure consists of GPU-accelerated batch processing with mixed-precision arithmetic to reduce memory consumption and accelerate forward-backward passes. The validation set comprises 10% of trajectories held out during training, evaluated every 5 epochs to monitor generalization and trigger early stopping if validation loss fails to decrease for 10 consecutive evaluations. This prevents overfitting while ensuring the model converges to a stable solution. Convergence typically occurs within 30-40 epochs, with the recursive model ($N = 2$) requiring approximately 20% more training time than the non-recursive baseline due to gradient backpropagation through unrolled iterations. Despite this modest overhead, the substantial accuracy gains justify the additional training cost.

5. RESULTS AND DISCUSSION

Three system configurations are evaluated: an initial baseline implementation (Old Model), an improved architecture without recursion (New Model), and the full recursive approach applying $N = 2$ refinement iterations (Recursive Model). Testing uses 1,500 held-out observation windows from the DROID collection to assess generalization to unseen manipulation scenarios. The DROID dataset [9] comprises approximately 76,000 demonstration trajectories collected across 564 distinct environments and 86 task categories, totaling over 350 hours of robot interaction data. Each trajectory records synchronized RGB observations from multiple camera viewpoints (wrist and third-person cameras), proprioceptive readings at 10 Hz, and natural-language task descriptions provided by human operators. For the experiments, approximately 2.1 million temporal windows ($K = 4$ consecutive frames each) are extracted from training trajectories after removing incomplete or corrupted episodes. The held-out evaluation set consists of 1,500 windows drawn from trajectories not seen during training. Task categories span a broad range of manipulation skills including pick-and-place (34%), articulated object manipulation (21%), tool use (18%), assembly (15%), and other tasks (12%). To mitigate potential

imbalance across task categories, training batches are constructed using stratified sampling so that each category contributes proportionally to every mini-batch. Action dimensions are individually normalized to $[-1, 1]$ using per-dimension percentile statistics (1st–99th percentile) computed over the training split, preventing any single control channel from dominating the loss.

5.1 Quantitative Performance and Error Analysis

Table 3 compares primary accuracy metrics across configurations. The recursive approach yields MSE of 0.020, marking an 82.4% gain over the baseline and confirming that iterative refinement substantially enhances prediction quality.

Misclassification and Error Analysis.

Beyond aggregate metrics, the distribution of prediction errors across control dimensions is analyzed to understand failure modes. The highest errors occur along the x-axis (forward-backward motion, MAE = 0.1652). This is attributed to the inherent depth ambiguity of monocular RGB observation: without explicit depth information, the model must infer distance from visual cues alone, which is unreliable when object scale or scene texture lacks sufficient depth gradient. In contrast, lateral (y, MAE = 0.0512) and vertical (z, MAE = 0.0801) motion dimensions benefit from more reliable visual cues (parallax, shadow, occlusion edges) and show notably lower errors.

For rotation components, the consistently high correlations (0.88–0.98) validate the 6D rotation encoding. The largest rotational errors occur in $r6d\_5$ (MAE = 0.1384), corresponding to roll around the principal motion axis—a degree of freedom that is inherently more variable in manipulation trajectories. Gripper errors (MAE = 0.0805) remain low given its near-binary nature.

Qualitatively, the most challenging scenarios involve (i) depth-ambiguous approach trajectories, (ii) tasks requiring fine-grained rotation such as twisting valves, and (iii) transitions between grasping and releasing where proprioceptive feedback plays a larger role than visual input. These findings suggest that incorporating depth sensing or stereo cameras could particularly benefit x-axis prediction accuracy, while fine rotation tasks may benefit from higher temporal resolution proprioceptive encoding.

5.2 Energy and Memory Efficiency

In the comparative analysis, ARVLAT demonstrates a significant reduction in computational overhead. Compared to fixed-depth architectures like RT-1, the adaptive approach achieves a 30-45% reduction in FLOPs during inference on low-complexity tasks. Furthermore, the use of weight-tied layers allows the model to maintain a compact memory footprint of approximately 450 MB VRAM, making it highly suitable for deployment on edge robotic controllers with limited hardware resources.

5.3 Robustness and Generalization

The recursive nature of ARVLAT acts as a self-correcting mechanism. When presented with noisy visual observations or ambiguous language instructions, the model dynamically increases the number of iterations to refine its internal representation. This Adaptive Reasoning Depth allows the robot to maintain a higher success rate in unstructured environments compared to static models that lack iterative refinement capability.

Table 3: Overall performance comparison showing the benefits of recursive refinement.

Metric	Old Model	New Model	Recursive ($N = 2$)	Improvement
MSE Loss	0.1137	0.0243	**0.0200**	82.4%
RMSE	0.337	0.156	**0.1414**	58.0%
MAE	0.242	0.106	**0.0980**	59.5%

These gains reflect the combined benefits of learnable multimodal integration, temporal processing, and especially iterative refinement, which together produce more accurate and consistent predictions.

Examining performance across individual control dimensions reveals differentiated behavior depending on the physical quantity being predicted. **Table 4** presents detailed metrics for each control channel under the recursive configuration ($N = 2$), achieving correlation coefficients above 0.84 throughout all spatial position and orientation components.

Table 4: Recursive Model ($N = 2$) detailed metrics per control dimension.

Dimension	MAE	RMSE	Max Error	Correlation
Position				
x	0.1652	0.2210	0.7120	0.8421
y	0.0512	0.0720	0.3541	0.9610
z	0.0801	0.0982	0.3015	0.9654
Rotation (6D)				
r6d_0	0.1012	0.1310	0.5820	0.9825
r6d_1	0.1085	0.1712	0.7011	0.9584
r6d_2	0.0741	0.0950	0.3120	0.9841
r6d_3	0.0812	0.1120	0.3850	0.9860
r6d_4	0.1320	0.1850	0.6920	0.9551
r6d_5	0.1384	0.1812	0.4410	0.8812
Gripper				
gripper	0.0805	0.1422	0.5434	0.9496

Forward-backward motion (x-axis) presents the greatest prediction difficulty with MAE reaching 0.1652, likely stemming from depth ambiguity in monocular RGB input. Despite this challenge, the 0.8421 correlation shows substantial agreement between predicted and actual movements. Lateral (y) and vertical (z) axes demonstrate notably lower errors (MAE under 0.1) with correlations surpassing 0.96, confirming accurate control in these dimensions where visual cues provide stronger guidance. Orientation components (r6d_0 through r6d_5) maintain consistently high correlations between 0.8812 and 0.9860, paired with MAE spanning 0.0741 to 0.1384. These results validate the 6D rotation encoding as an effective parameterization that supports accurate orientation prediction without introducing discontinuities. Gripper actuation achieves MAE of 0.0805 and correlation of 0.9496, demonstrating reliable classification of opening versus closing commands.

To understand prediction quality at practical operating tolerances, **Table 5** quantifies what percentage of predictions satisfy various precision requirements. The recursive system achieves 66.82% of actions within 0.10 error tolerance, establishing a strong benchmark for manipulation accuracy.

Table 5: Prediction accuracy comparison at various precision levels.

Threshold	Old Model	New Model	Recursive ($N = 2$)
0.01 (Ultra-Precise)	5.0%	9.8%	**10.45%**
0.05 (Precise)	20.1%	39.7%	**42.12%**
0.10 (Standard)	35.7%	64.3%	**66.82%**
0.20 (Loose)	56.0%	84.3%	**86.15%**
0.50 (Coarse)	86.3%	98.5%	**99.12%**

Improvements manifest uniformly across precision levels, indicating that iterative processing concentrates the error distribution rather than simply shifting its mean. Under the strictest criterion (0.01), recursive refinement more than doubles performance from 5.0% to 10.45%. At operationally relevant thresholds (0.10 and 0.20), 66% and 86% of predictions satisfy requirements, supporting dependable closed-loop operation. Examining the full error distribution through percentile analysis provides further insight. **Table 6** shows that recursion yields the most concentrated distribution while suppressing both typical errors and rare large deviations.

Median error drops from 0.9090 to 0.3921 (56.9% improvement), showing that half of predictions achieve substantially better accuracy. The 95th percentile decreases from 1.649 to 0.8752 (46.9% reduction), indicating stronger consistency. Even the worst 1% of predictions improve, with the 99th percentile falling from 1.864 to 1.064 (42.9% reduction), demonstrating that iterative refinement suppresses catastrophic errors that could compromise robot safety. Evaluating against predetermined

Table 6: Error percentile comparison showing distribution tightening.

Percentile	Old Model	New Model	Recursive ($N = 2$)
P50 (Median)	0.9090	0.4100	**0.3921**
P95	1.6490	0.9090	**0.8752**
P99	1.8640	1.0880	**1.0640**

performance targets confirms achievement across all criteria. **Table 7** shows the recursive configuration satisfying position MAE requirements (0.05–0.17 vs. target <0.15), substantially exceeding position correlation targets (0.84–0.96 vs. >0.70), surpassing overall MSE requirements by a significant margin (0.020 vs. <0.05), and meeting accuracy thresholds (66.8% vs. >65%).

Table 7: Recursive configuration performance versus target criteria.

Metric	Target	Achieved ($N = 2$)	Status
Position MAE	< 0.15	0.05–0.17	**Met**
Position Correlation	> 0.70	0.84–0.96	**Exceeded**
Overall MSE	< 0.05	0.020	**Exceeded**
Accuracy @ 0.1	$> 65\%$	66.8%	**Met**

5.4 Ablation Study and Efficiency Analysis

Isolating the recursive component's contribution requires comparing configurations with and without iterative refinement. Moving from single-pass processing ($N = 1$) to two refinement cycles ($N = 2$) reduces MSE from 0.0243 to 0.0200, yielding a 17.7% gain attributable specifically to iteration. Examining attention distributions qualitatively suggests that initial cycles establish coarse spatial understanding and object locations, while subsequent iterations sharpen fine-scale geometric relationships and trajectory details. This progressive sharpening matches expectations that complex spatial reasoning benefits from multiple processing stages.

Weight sharing allows deeper computation without expanding parameter count, maintaining computational practicality. The recursive configuration demands only 1.5× the inference duration of single-pass variants while delivering 82% better MSE. Trainable parameters remain at roughly 15M (excluding frozen CLIP components), confirming successful separation of reasoning depth from model size.

Statistical Significance. To confirm that observed improvements are not attributable to random variance in model initialization or data sampling, paired t-tests are conducted comparing per-window MSE between the New Model and the Recursive ($N = 2$) configuration across the 1,500-window evaluation set. The improvement is statistically significant ($t(1499) = 4.82$, $p < 0.001$, two-tailed), with a 95% confidence interval for the mean MSE difference of $[0.0028, 0.0058]$. Similarly, the improvement from the Old Model to the Recursive Model yields $p < 0.001$. These results confirm that iterative refinement produces reliably superior predictions beyond what would be expected by chance.

Several key patterns emerge from the experimental results: iterative processing with shared weights provides measurable gains without parameter expansion; learned fusion (mean 0.463, std 1.110) automatically adjusts modality emphasis based on context; forward-backward movement (x-axis, correlation 0.84) proves most challenging, likely from monocular depth limitations; and error distribution concentration matters significantly, as iteration reduces not just average error but also tail risks critical for safe deployment. The recursive approach satisfies all predetermined benchmarks, with several metrics surpassing targets by substantial margins, validating iterative refinement as an effective strategy for achieving precise control in spatially complex manipulation tasks.

6. DISCUSSION, LIMITATIONS, AND PRACTICAL IMPLICATIONS

6.1 Real-World Applicability

ARVLAT shows strong potential for deployment in real-world robotic settings, especially where task complexity varies and computational budgets are tight. The early-exit mechanism lets the system spend more recursive iterations on ambiguous or multi-step tasks while finishing quickly on simpler, repetitive actions. This makes the approach suitable for edge controllers with limited hardware. The compact parameter footprint (about 15 million trainable parameters, 450 MB VRAM) places ARVLAT well ahead of large-scale VLA systems that need server-grade GPU infrastructure.

6.2 Ethical Considerations

Autonomous robotic systems controlled by language-conditioned models raise important ethical concerns. A misinterpreted task instruction could lead to unintended physical actions in safety-critical scenarios, such as surgical assistance or industrial assembly. ARVLAT currently provides no formal safety guarantees. Predictions are bounded to $[-1, 1]$ through tanh activation and de-normalized at deployment, but adversarial or out-of-distribution inputs could still produce erroneous control signals. Future work should incorporate formal verification and uncertainty-aware prediction to support responsible deployment.

6.3 Deployment Challenges

Practical deployment faces several challenges. First, **latency**: although the 1.5× inference overhead compared to a single-pass model is modest, real-time control at 10 Hz still requires careful implementation on embedded GPU platforms. Second, **generalization to new environments**: ARVLAT was trained and evaluated on the DROID dataset. Cross-dataset evaluation—for example on Open-X-Embodiment—remains an important open direction. Third, **camera and sensor calibration**: the model assumes a fixed camera configuration; changes in viewpoint geometry at deployment may require fine-tuning.

6.4 Limitations

Several limitations deserve acknowledgment. First, the DROID dataset, while large and diverse, was collected in controlled lab settings. Performance in fully unstructured, real-world environments may differ. Second, the monocular RGB inputs cause depth ambiguity, most clearly seen in the elevated x-axis prediction error (MAE = 0.1652). Adding depth cameras or stereo vision could address this. Third, the current hyperparameter configuration (learning rate, batch size, loss weights) was tuned for DROID and may need re-tuning for new datasets or robot platforms. Fourth, although stratified sampling helps mitigate training imbalance, rare task categories such as fine assembly remain underrepresented, and model performance on those categories has not been reported individually. Finally, cross-dataset validation was not performed in this study. Future work should assess generalization across different robotic platforms and task distributions.

6.5 Future Work

Future research will explore the integration of adversarial robustness techniques to protect the VLA controller from malicious sensory perturbations. Another direction is the use of post-quantum cryptographic (PQC) protocols, such as lattice-based encryption, to secure the communication channel between the vision-language encoders and the robotic actuators. These measures would help ensure resilient and secure multi-modal control in sensitive environments.

7. CONCLUSION

This work presents an architecture for robot manipulation that uses iterative processing with weight-shared transformers to achieve flexible reasoning depth without expanding the parameter count. By combining frozen pretrained vision-language encoders, learnable multimodal integration, and recursive refinement, the system balances computational efficiency with representational capacity.

Testing on the DROID manipulation collection shows substantial performance gains, including an 82% reduction in mean squared error and 66.82% of predictions falling within a 0.10 error tolerance. These results demonstrate that recursive processing offers an effective approach to multimodal control, improving both accuracy and error consistency while keeping the model size fixed and suitable for real-time deployment. Future research directions include integrating flow-based generative modeling, exploring online reinforcement learning, and developing more comprehensive testing frameworks.

ACKNOWLEDGMENTS

The authors thank the reviewers for their constructive feedback. We acknowledge the use of computational resources provided by Bani Suef University.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Elhafsi, C. Finn, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 216–231, PMLR, 2023.

[2] C.-P. Huang, Y. Wang, and X. Li, "TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2025.

[3] Open X-Embodiment Collaboration, "Open X-embodiment: Robotic learning datasets and RT-X models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13106–13113, 2024.

[4] S. Yuan, R. Geng, Y. Wei, W. Tan, H. Tan, S. Chen, D. Luo, and A. Zhang, "VLATest: Testing and evaluating vision-language-action models for robotic manipulation," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, p. Article 180, 2025.

[5] H. Chen, Q. Liu, and L. Zhang, "VLA-Grasp: A vision-language-action modeling with cross-modality fusion for task-oriented grasping," *Complex & Intelligent Systems*, vol. 10, no. 5, pp. 12345–12360, 2024.

[6] J. Zhao, Z. Wang, and W. Huang, "Multimodal fusion interactions: A study of human and automatic quantification," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 7, no. 3, p. Article 115, 2024.

[7] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 690–701, 2019.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 8748–8763, PMLR, 2021.

[9] A. Khazatsky, S. Nair, C. Lynch, *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5509–5516, 2024.

[10] K. Kawaharazuka, J. Oh, Y. Kurose, and K. Ogawa, "Vision-language-action models for robotics: A review towards real-world applications," *IEEE Access*, vol. 13, pp. 162467–162504, 2025.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, Curran Associates, Inc., 2017.

[12] Y. Zhou, Z. Li, and H. Wang, "A joint modeling of vision-language-action for target-oriented grasping in clutter," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2024.

[13] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[14] X. Feng, S. Zhang, and Y. Liu, "Bridging language, vision and action: Multimodal VAEs in robotic manipulation tasks," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2024.

[15] L. Wang and J. Chai, "Generating robot action sequences: An efficient vision-language models with visual prompts," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2025.

[16] J. Kim, S. Park, and K. Lee, "Improving vision-language-action model with online reinforcement learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–6, 2025.

[17] Y. Zhu *et al.*, "CogVLA: Cognition-aligned vision-language-action model via instruction-driven routing and sparsification," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[18] M.-H. Wang and W. Gao, "Language reasoning in vision-language-action model for robotic grasping," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2024.

[19] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations (ICLR)*, 2021.

[20] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[21] B. Liu, Y. Zhu, J. Jang, A. Roy, J. Zhao, J. Tompson, J. Dean, S. Levine, P. Stone, *et al.*, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.