

VLAF-AR: Video-Language-Action Fusion with Contrastive Alignment and Autoregressive Interleaving

A Unified Architecture for Language-Conditioned Robot Learning

Abstract

We propose **VLAF-AR**, a novel architecture for language-conditioned robot manipulation that unifies three key innovations: (1) **VLA-CLIP**, a trimodal contrastive pretraining objective that aligns video, language, and action representations in a shared embedding space; (2) **Autoregressive Interleaving**, which processes interleaved video-language token sequences through a causal transformer for dynamic cross-modal reasoning; and (3) a **three-stage training pipeline** analogous to modern LLM training (contrastive pretraining $\rightarrow$ interleaved behavioral cloning $\rightarrow$ RL fine-tuning).

Our key insight is that cross-modal fusion for robotics requires both **representational alignment** (knowing that “pick up red cup” and the corresponding video and action trajectory are semantically equivalent) and **sequential reasoning** (understanding how language tokens relate to specific video frames and action phases). VLA-CLIP provides the former through contrastive learning; autoregressive interleaving provides the latter through causal attention.

We hypothesize VLAF-AR will achieve state-of-the-art performance on complex manipulation tasks requiring fine-grained video-language-action alignment, while maintaining sample efficiency with only 1K-10K robot demonstrations and enabling zero-shot action retrieval via the learned embedding space.

1. Introduction

1.1 Problem Statement

Language-conditioned robot manipulation requires bridging three modalities:

- **Language:** Semantic understanding of goals (“pick up the cup behind the red block”)
- **Video:** Visual perception of scenes and dynamics (objects, spatial relations, motion)
- **Action:** Motor commands that achieve the goal (end-effector trajectories)

Current approaches handle at most two modalities well:

Approach	V-L Alignment	V-A Alignment	L-A Alignment	Cross-Modal Reasoning
CLIP				Static
VLA				Implicit
(RT-2)				
VAM				Temporal
(Cosmos)				
VLAF				Static fusion
(Gated)				

None provide both **explicit trimodal alignment** and **dynamic cross-modal reasoning**.

1.2 Key Insights

Insight 1: Contrastive Alignment as Foundation

Before learning to *reason* across modalities, the model should understand that video of “picking up a red cup”, the phrase “pick up the red cup”, and the corresponding action trajectory are *the same concept* in different forms. Contrastive learning provides this foundation.

Insight 2: Sequential Reasoning via Autoregression

Understanding “the cup *behind* the red block” requires iterative attention: process “cup” → attend to cup candidates in video → process “behind” → filter by spatial relation → process “red block” → finalize grounding. Autoregressive modeling makes this explicit.

Insight 3: LLM Training Pipeline for Robotics

Modern LLMs succeed via staged training: pretraining → SFT → RLHF. We adapt this for robotics: contrastive pretraining → interleaved BC → RL fine-tuning.

1.3 Contributions

1. **VLA-CLIP**: A trimodal contrastive objective aligning video, language, and action in a shared embedding space
2. **Autoregressive Interleaving**: Dynamic cross-modal reasoning via causal attention over interleaved sequences
3. **Unified Training Pipeline**: Four-stage approach from contrastive pretraining to RL fine-tuning

4. **Zero-Shot Capabilities:** Action retrieval and cross-task transfer via the learned embedding space
-

2. Method

2.1 Overview

VLA-AR consists of two phases:

Phase 1: VLA-CLIP Pretraining — Learn a shared embedding space for video, language, and action via contrastive learning

Phase 2: VLA-AR Training — Train an autoregressive transformer over interleaved tokens from the frozen VLA-CLIP encoders

2.2 VLA-CLIP: Trimodal Contrastive Pretraining

2.2.1 Encoders

We define three modality-specific encoders:

Video Encoder f_v : Maps video frames to a sequence of tokens, then pools to a single vector:

$$h_v = \text{Pool}(f_v(\mathbf{v}_{1:T})) \in \mathbb{R}^{d_v}$$

Architecture: Cosmos-Predict2 DiT or VideoMAE

Language Encoder f_ℓ : Maps token sequence to a single vector:

$$h_\ell = \text{Pool}(f_\ell(\ell)) \in \mathbb{R}^{d_\ell}$$

Architecture: Llama-3 or CLIP text encoder

Action Encoder f_a : Maps action trajectory to a single vector:

$$h_a = f_a(a_{1:k}) \in \mathbb{R}^{d_a}$$

Architecture: Transformer or 1D CNN over action chunks

2.2.2 Projection Heads

Project each modality to a shared d -dimensional space:

$$z_v = \text{Proj}_v(h_v) = \frac{W_v h_v + b_v}{\|W_v h_v + b_v\|_2} \in \mathbb{R}^d$$

$$z_\ell = \text{Proj}_\ell(h_\ell) = \frac{W_\ell h_\ell + b_\ell}{\|W_\ell h_\ell + b_\ell\|_2} \in \mathbb{R}^d$$

$$z_a = \text{Proj}_a(h_a) = \frac{W_a h_a + b_a}{\|W_a h_a + b_a\|_2} \in \mathbb{R}^d$$

L2 normalization ensures embeddings lie on a unit hypersphere.

2.2.3 Contrastive Objectives

We train with three pairwise InfoNCE losses:

Video-Language Alignment:

$$\mathcal{L}_{V-L} = -\frac{1}{2B} \sum_{i=1}^B \left[\log \frac{\exp(z_{v,i}^\top z_{\ell,i}/\tau)}{\sum_{j=1}^B \exp(z_{v,i}^\top z_{\ell,j}/\tau)} + \log \frac{\exp(z_{\ell,i}^\top z_{v,i}/\tau)}{\sum_{j=1}^B \exp(z_{\ell,i}^\top z_{v,j}/\tau)} \right]$$

Video-Action Alignment:

$$\mathcal{L}_{V-A} = -\frac{1}{2B} \sum_{i=1}^B \left[\log \frac{\exp(z_{v,i}^\top z_{a,i}/\tau)}{\sum_{j=1}^B \exp(z_{v,i}^\top z_{a,j}/\tau)} + \log \frac{\exp(z_{a,i}^\top z_{v,i}/\tau)}{\sum_{j=1}^B \exp(z_{a,i}^\top z_{v,j}/\tau)} \right]$$

Language-Action Alignment:

$$\mathcal{L}_{L-A} = -\frac{1}{2B} \sum_{i=1}^B \left[\log \frac{\exp(z_{\ell,i}^\top z_{a,i}/\tau)}{\sum_{j=1}^B \exp(z_{\ell,i}^\top z_{a,j}/\tau)} + \log \frac{\exp(z_{a,i}^\top z_{\ell,i}/\tau)}{\sum_{j=1}^B \exp(z_{a,i}^\top z_{\ell,j}/\tau)} \right]$$

Total VLA-CLIP Loss:

$$\boxed{\mathcal{L}_{VLA-CLIP} = \mathcal{L}_{V-L} + \mathcal{L}_{V-A} + \mathcal{L}_{L-A}}$$

2.2.4 What VLA-CLIP Learns

The contrastive objectives ensure:

Property	Meaning
$z_v \approx z_\ell$	Video and its description are nearby
$z_v \approx z_a$	Video and action that produced it are nearby
$z_\ell \approx z_a$	Instruction and action that achieves it are nearby
$z_v \approx z_\ell \approx z_a$	Full trajectory is a coherent point in space

This enables **zero-shot action retrieval**: given a new instruction ℓ , find the nearest action in embedding space:

$$a^* = \arg \max_{a \in \mathcal{A}_{\text{database}}} z_\ell^\top z_a$$

2.3 Autoregressive Interleaving

After VLA-CLIP pretraining, we freeze the encoders and train an autoregressive transformer.

2.3.1 Token Extraction

Instead of pooling to single vectors, we extract token sequences from frozen encoders:

$$\begin{aligned} z_v^{\text{seq}} &= f_v(\mathbf{v}_{1:T}) \in \mathbb{R}^{M \times d} \\ z_\ell^{\text{seq}} &= f_\ell(\ell) \in \mathbb{R}^{N \times d} \end{aligned}$$

These tokens are already in the aligned VLA-CLIP embedding space (via the projection heads applied per-token).

2.3.2 Interleaved Sequence Construction

We construct the input sequence by interleaving video and language tokens:

$$x = [v_1, \ell_1, v_2, \ell_2, \dots, v_M, \ell_N, [\text{ACT}], p]$$

Where: - $v_i \in \mathbb{R}^d$: Video token i (from frozen VLA-CLIP encoder) - $\ell_j \in \mathbb{R}^d$: Language token j (from frozen VLA-CLIP encoder) - $[\text{ACT}] \in \mathbb{R}^d$: Learned action query token - $p \in \mathbb{R}^d$: Proprioceptive embedding

Modality Embeddings:

$$\tilde{x}_i = x_i + e_{\text{pos}}(i) + e_{\text{mod}}(m_i)$$

Where $m_i \in \{\text{video}, \text{language}, \text{action}, \text{proprio}\}$.

2.3.3 Autoregressive Transformer

The AR transformer processes the sequence with causal self-attention:

$$h_t = \text{Transformer}(x_{\leq t})$$

Attention Computation:

$$\text{Attn}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} + M_{\text{causal}} \right) V$$

Where M_{causal} is $-\infty$ for future positions and 0 otherwise.

Key Property: Because tokens are in the shared VLA-CLIP space, cross-modal attention is meaningful from the start. A language token for “red” can directly attend to video tokens of red objects because they have similar embeddings.

2.3.4 Action Decoder

The action decoder maps $h_{[\text{ACT}]}$ to action chunks:

MLP Decoder:

$$a_{t:t+k} = \text{MLP}(h_{[\text{ACT}]}) \in \mathbb{R}^{k \times d_a}$$

Flow Matching Decoder:

$$a = a_0 + \int_0^1 v_\theta(a_t, t, h_{[\text{ACT}]}) dt, \quad a_0 \sim \mathcal{N}(0, I)$$

2.4 Training Objectives (Phase 2)

2.4.1 Action Prediction Loss

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{(\mathbf{v}, \ell, p, a^*) \sim \mathcal{D}} [\|\pi_\theta(h_{[\text{ACT}]}) - a^*\|^2]$$

2.4.2 Autoregressive Next-Token Loss

$$\mathcal{L}_{\text{AR}} = - \sum_{t=1}^{|x|-1} \log p_\theta(x_{t+1} | h_t)$$

Since video and language tokens are in the same VLA-CLIP space, we can use a unified prediction head:

$$p_\theta(x_{t+1} | h_t) \propto \exp(h_t^\top x_{t+1} / \tau)$$

This is essentially **contrastive next-token prediction**: pull h_t toward the true next token and push away from other tokens in the batch.

2.4.3 Auxiliary Video Prediction Loss (Optional)

$$\mathcal{L}_{\text{aux}} = \|\text{Decode}(h_{v_t}) - v_{t+1}\|^2$$

2.4.4 Total Loss

$$\mathcal{L}_{\text{VLA-AR}} = \mathcal{L}_{\text{action}} + \lambda_1 \mathcal{L}_{\text{AR}} + \lambda_2 \mathcal{L}_{\text{aux}}$$

2.5 Full Training Pipeline

Stage	Name	Loss	Trainable	Data	Analogy
0	VLA-CLIP Pretraining	$\mathcal{L}_{\text{VLA-CLIP}}$	Encoders + Projections	Robot demos + video-text pairs	CLIP pretraining
1	Freeze Encoders	—	—	—	—
2	Interleaved BC	$\mathcal{L}_{\text{action}} + \lambda_1 \mathcal{L}_{\text{AR}}$	AR Transformer + Action Decoder	Open X-Embodiment	SFT / Mid-training
3	RL Fine-tuning	$J(\theta) = \mathbb{E}[\sum \gamma^t r_t]$	Action Decoder (+ LoRA)	Target task rollouts	RLHF / RLVR

3. Architecture Details

3.1 VLA-CLIP Encoders

Component	Architecture	Parameters	Output
Video Encoder	Cosmos-Predict2 DiT	~300M	M tokens, $d = 1024$
Language Encoder	Llama-3-8B (frozen base)	~8B	N tokens, $d = 1024$
Action Encoder	4-layer Transformer	~10M	1 token, $d = 1024$
Projection Heads	2-layer MLP	~2M each	$d = 512$ (shared space)

3.2 AR Transformer

Hyperparameter	Value
Layers	12
Hidden dim	1024
Attention heads	16
FFN dim	4096
Dropout	0.1
Max sequence length	512

3.3 Action Decoder

Component	Architecture
MLP Decoder	3-layer MLP, hidden=512, ReLU
Flow Matching	6-layer Transformer, 100 diffusion steps
Action chunk size	$k = 8$
Action dim	$d_a = 7$ (6-DoF + gripper)

4. Theoretical Analysis

4.1 Why VLA-CLIP Helps Autoregressive Training

Without VLA-CLIP: The AR transformer receives tokens from different embedding spaces. Cross-modal attention must learn alignment from scratch.

With VLA-CLIP: Tokens are pre-aligned. “Red” (language) and red pixels (video) have high cosine similarity before AR training begins.

Formally, let $\text{sim}(v_i, \ell_j) = v_i^\top \ell_j$ be the pre-AR attention affinity. VLA-CLIP ensures:

$$\mathbb{E}[\text{sim}(v_i, \ell_j) | \text{semantically related}] > \mathbb{E}[\text{sim}(v_i, \ell_j) | \text{unrelated}]$$

This provides a strong initialization for the attention patterns.

4.2 Information-Theoretic View

VLA-CLIP maximizes mutual information between modalities:

$$\mathcal{L}_{\text{VLA-CLIP}} \propto -I(Z_v; Z_\ell) - I(Z_v; Z_a) - I(Z_\ell; Z_a)$$

The AR objective then captures *conditional* dependencies:

$$\mathcal{L}_{\text{AR}} \propto -I(X_{t+1}; X_{\leq t})$$

Together, they capture both **global alignment** (VLA-CLIP) and **local sequential dependencies** (AR).

4.3 Comparison with Alternatives

Method	Global Alignment	Local Reasoning	Computation
CLIP + MLP	(V-L only)		$O(M + N)$
Gated Fusion	Learned gate	Static	$O(M + N)$
Cross-Attention	Implicit	Bidirectional	$O(MN)$
VLA-CLIP + AR	(V-L-A)	Causal	$O((M + N)^2)$

4.4 Zero-Shot Capabilities

VLA-CLIP enables several zero-shot capabilities:

Action Retrieval: Given instruction ℓ , find nearest action:

$$a^* = \arg \max_{a \in \mathcal{A}} z_\ell^\top z_a$$

Video-to-Action: Given video $\mathbf{v}$, find corresponding action:

$$a^* = \arg \max_{a \in \mathcal{A}} z_v^\top z_a$$

Cross-Task Transfer: Tasks with similar language descriptions have similar action embeddings, enabling transfer.

5. Experimental Plan

5.1 Datasets

VLA-CLIP Pretraining: - Open X-Embodiment (~1M robot trajectories) - Ego4D (egocentric video, no actions) - WebVid (video-text pairs, no actions)

Interleaved BC: - Open X-Embodiment - Bridge V2 - CALVIN

RL Fine-tuning: - Target task environment (simulated or real)

5.2 Benchmarks

Benchmark	Focus	Metrics
CALVIN	Long-horizon, language-conditioned	Success rate, chain length
Language-Table	Spatial reasoning	Success rate by relation type
RLBench	Diverse manipulation	Per-task success rate
Real Franka	Sim-to-real transfer	Success rate, robustness

5.3 Baselines

- **OpenVLA:** VLA baseline
- **RT-2:** Large VLA
- **Octo:** Diffusion policy
- **VLA^F (Gated):** Gated fusion (no AR)
- **VLA^F-AR (no CLIP):** AR interleaving without VLA-CLIP pretraining

5.4 Ablations

Ablation	Purpose
Remove $\mathcal{L}_{V-L}$	Importance of video-language alignment
Remove $\mathcal{L}_{V-A}$	Importance of video-action alignment
Remove $\mathcal{L}_{L-A}$	Importance of language-action alignment
Remove $\mathcal{L}_{AR}$	Importance of autoregressive objective
Concatenate vs. Interleave	Importance of interleaving strategy
Vary λ_1	AR loss weight sensitivity

6. Hypotheses

H1 (Trimodal Alignment): VLA-CLIP pretraining will improve downstream performance by 15% compared to training without contrastive alignment.

H2 (Complex Reasoning): VLAF-AR will outperform gated fusion by 10% on tasks with complex spatial references (“behind”, “between”, “next to”).

H3 (Sample Efficiency): VLA-CLIP + AR will achieve equivalent performance with 50% fewer demonstrations due to better initialization.

H4 (Zero-Shot Retrieval): VLA-CLIP embeddings will enable zero-shot action retrieval with 60% accuracy on held-out instructions.

H5 (Cross-Modal Grounding): Attention visualization will show interpretable patterns (e.g., “red” attending to red pixels).

H6 (Latency Tradeoff): VLAF-AR will have higher latency (~150ms) than gated fusion (~50ms) but better accuracy on complex tasks.

7. Implementation Details

7.1 VLA-CLIP Training

```
# Pseudocode for VLA-CLIP forward pass
def vla_clip_forward(video, language, action):
    # Encode
    h_v = video_encoder(video)      # [B, d_v]
    h_l = language_encoder(language) # [B, d_l]
    h_a = action_encoder(action)    # [B, d_a]

    # Project and normalize
    z_v = F.normalize(proj_v(h_v), dim=-1) # [B, d]
    z_l = F.normalize(proj_l(h_l), dim=-1) # [B, d]
    z_a = F.normalize(proj_a(h_a), dim=-1) # [B, d]
```

```

# Compute pairwise logits
logits_vl = z_v @ z_l.T / tau # [B, B]
logits_va = z_v @ z_a.T / tau # [B, B]
logits_la = z_l @ z_a.T / tau # [B, B]

# InfoNCE losses
labels = torch.arange(B)
loss_vl = (F.cross_entropy(logits_vl, labels) +
           F.cross_entropy(logits_vl.T, labels)) / 2
loss_va = (F.cross_entropy(logits_va, labels) +
           F.cross_entropy(logits_va.T, labels)) / 2
loss_la = (F.cross_entropy(logits_la, labels) +
           F.cross_entropy(logits_la.T, labels)) / 2

return loss_vl + loss_va + loss_la

```

7.2 Interleaved AR Training

```

# Pseudocode for VLAF-AR forward pass
def vlaf_ar_forward(video, language, proprio, action_gt):
    # Get token sequences from frozen VLA-CLIP encoders
    z_v = frozen_video_encoder(video) # [B, M, d]
    z_l = frozen_language_encoder(language) # [B, N, d]
    z_p = proprio_embed(proprio) # [B, 1, d]
    act_token = act_query.expand(B, 1, d) # [B, 1, d]

    # Interleave
    x = interleave(z_v, z_l) # [B, M+N, d]
    x = torch.cat([x, act_token, z_p], dim=1) # [B, M+N+2, d]

    # AR transformer with causal mask
    h = ar_transformer(x) # [B, M+N+2, d]

    # Action prediction from [ACT] position
    h_act = h[:, -2, :] # [B, d]
    a_pred = action_decoder(h_act) # [B, k, d_a]

    # Losses

```

```

loss_action = F.mse_loss(a_pred, action_gt)
loss_ar = ar_next_token_loss(h[:, :-1], x[:, 1:])

return loss_action + lambda1 * loss_ar

```

7.3 Hyperparameters

Parameter	VLA-CLIP	VLAF-AR
Batch size	2048	256
Learning rate	1e-4	1e-4
Optimizer	AdamW	AdamW
Weight decay	0.1	0.01
Warmup steps	10,000	1,000
Temperature τ	0.07	0.1
λ_1 (AR loss)	—	0.1
λ_2 (aux loss)	—	0.01
Training steps	500K	100K

8. Expected Results

8.1 Main Results (Projected)

Method	CALVIN (Avg Len)	Language-Table	RLBench (Avg)
OpenVLA	2.1	68%	72%
RT-2	2.4	74%	78%
Octo	1.8	62%	70%
VLAF (Gated)	2.6	76%	80%
VLAF-AR (no CLIP)	2.8	79%	82%
VLAF-AR (full)	3.2	85%	87%

8.2 Ablation Results (Projected)

Ablation	CALVIN	Δ
Full VLAF-AR	3.2	—
– VLA-CLIP pretraining	2.8	-12.5%
– AR loss	2.9	-9.4%
– V-A contrastive	3.0	-6.3%
– L-A contrastive	3.0	-6.3%
Concatenate (no interleave)	2.7	-15.6%

9. Limitations and Future Work

9.1 Limitations

1. **Compute Cost:** $(M + N)^2$ attention may limit real-time deployment
2. **Data Requirements:** VLA-CLIP pretraining benefits from large-scale paired data
3. **Modality Gap:** Despite contrastive learning, some modality gap may remain

9.2 Future Directions

1. **Sparse Attention:** Reduce complexity to $O((M + N) \log(M + N))$
 2. **Distillation:** Compress trained model for real-time inference
 3. **Additional Modalities:** Add tactile, audio, depth
 4. **Hierarchical VLA-CLIP:** Frame-level + trajectory-level contrastive
 5. **Self-Supervised Actions:** Learn action encoder without labeled actions
-

10. Conclusion

VLAF-AR presents a principled approach to unifying video, language, and action representations for robot learning. By combining **trimodal contrastive pretraining** (VLA-CLIP) with **autoregressive interleaving**, we achieve both representational alignment and dynamic cross-modal reasoning.

The key innovations are:

1. **VLA-CLIP:** First trimodal (V-L-A) contrastive objective for robotics

2. **Aligned AR:** Autoregressive transformer over pre-aligned tokens
3. **LLM-style Pipeline:** Staged training from pretraining to RL fine-tuning
4. **Zero-Shot Capabilities:** Action retrieval via embedding similarity

We hypothesize VLAF-AR will advance the state-of-the-art on language-conditioned manipulation, particularly for tasks requiring complex spatial reasoning, multi-step planning, and cross-modal grounding.

References

1. Radford et al. “Learning Transferable Visual Models From Natural Language Supervision.” ICML 2021.
 2. Brohan et al. “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control.” CoRL 2023.
 3. Kim et al. “OpenVLA: An Open-Source Vision-Language-Action Model.” 2024.
 4. NVIDIA. “Cosmos: World Foundation Models for Physical AI.” 2024.
 5. Black et al. “ : A Vision-Language-Action Flow Model for General Robot Control.” 2024.
 6. Nair et al. “R3M: A Universal Visual Representation for Robot Manipulation.” CoRL 2022.
 7. Ma et al. “VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training.” ICLR 2023.
-

Appendix A: Loss Function Summary

A.1 VLA-CLIP (Stage 0)

$$\mathcal{L}_{\text{VLA-CLIP}} = \underbrace{-\log \frac{e^{z_v^\top z_\ell / \tau}}{\sum_j e^{z_v^\top z_{\ell,j} / \tau}}}_{\mathcal{L}_{\text{V-L}}} + \underbrace{-\log \frac{e^{z_v^\top z_a / \tau}}{\sum_j e^{z_v^\top z_{a,j} / \tau}}}_{\mathcal{L}_{\text{V-A}}} + \underbrace{-\log \frac{e^{z_\ell^\top z_a / \tau}}{\sum_j e^{z_\ell^\top z_{a,j} / \tau}}}_{\mathcal{L}_{\text{L-A}}} + (\text{symmetric terms})$$

A.2 VLAF-AR (Stage 2)

$$\mathcal{L}_{\text{VLAF-AR}} = \underbrace{\|\pi_\theta(h_{[\text{ACT}]}) - a^*\|^2}_{\mathcal{L}_{\text{action}}} + \lambda_1 \underbrace{\left(-\sum_t \log p_\theta(x_{t+1} | h_t) \right)}_{\mathcal{L}_{\text{AR}}} + \lambda_2 \underbrace{\|\hat{v}_{t+1} - v_{t+1}\|^2}_{\mathcal{L}_{\text{aux}}}$$

A.3 RL Fine-tuning (Stage 3)

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^H \gamma^t r(s_t, a_t) \right], \quad r \in \{0, 1\}$$

Appendix B: Comparison Table

Method	V-L Align	V-A Align	L-A Align	Cross-Modal	Temporal	Zero-Shot
CLIP				Static		V L
R3M				Implicit		
VIP				Implicit		
RT-2				Implicit		Limited
VLAF				Static		
(Gated)						
VLAF- AR				Dynamic		V L A