WHEN THE GATE DOES NOT GATE: BOUNDED MEMORY INJECTION WITH NORMALIZED DIRECTION FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory-dependent manipulation requires robots to act on visual cues that are no longer observable. Existing memory-augmented vision-language-action (VLA) models retrieve history from an external memory bank and add it to the current representation as a gated residual. The gate is expected to control how strongly memory is injected. However, we show that this expectation often fails. Across common designs, the gate degenerates. It either saturates or collapses toward zero, losing its variation across inputs. Control over injection strength then shifts to the memory projection, which undermines the fusion of memory. We trace this to scale coupling between the gate and the projection. Because training constrains only their product, the gate value neither reflects nor bounds injection strength. We propose BIND-VLA, which decouples the direction and magnitude of the memory update. The projection is orthogonalized against the current representation and normalized, so it determines only the update direction. The gate sets the fraction of a fixed update budget. It thus becomes identifiable, and memory injection is explicitly bounded. BIND-VLA also compresses each past timestep into a single memory token and keeps history outside the backbone, leaving the input length unchanged. On nine MIKASA-Robo memory tasks, BIND-VLA reaches 38.4% average success, 6.0 points above dot-product gated fusion with the same architecture. Its gate no longer degenerates, with a coefficient of variation of about 50% compared with below 5% for the dot-product gate. Fixing it to its mean value lowers average success by 9.2 points. On LIBERO, the same gate switches memory off when history is not needed, and BIND-VLA matches the memory-free policy at 92.4% versus 92.2%.

1 INTRODUCTION

Vision-language-action (VLA) models map visual observations and language instructions directly to robot actions (Zitkovich et al., 2023; Kim et al., 2025b; Black et al., 2025; Physical Intelligence et al., 2025). Built on pretrained vision-language backbones, they generalize strongly in language-conditioned manipulation. Most of them, however, condition only on the current observation. Many manipulation tasks are inherently history-dependent. The cue needed for the current decision, such as the color, shape, or initial position of a target, may appear only briefly in early observations. By the time the robot acts on it, the cue may already be occluded or out of view. A policy that sees only the present cannot recover this information. Solving such tasks requires retaining task-relevant visual evidence over time and using it for later actions (Chen et al., 2026; Yang et al., 2026).

A direct way to provide history is to append past frames to the policy input (Koo et al., 2026). The cost of self-attention then grows with the history length, and long histories quickly become impractical. Memory-augmented VLAs instead store past observations in an external memory bank (Wang et al., 2026; Sun et al., 2026; Shi et al., 2026b;a). The current representation retrieves relevant history through attention and adds it back as a residual, scaled by a context-dependent gate. This design separates two decisions. Retrieval decides what history to use, and the gate decides how much of it to inject (Alayrac et al., 2022; Zhang et al., 2024). The separation matters because many steps do not need history, and injecting memory there can only perturb the pretrained representation (Jang

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

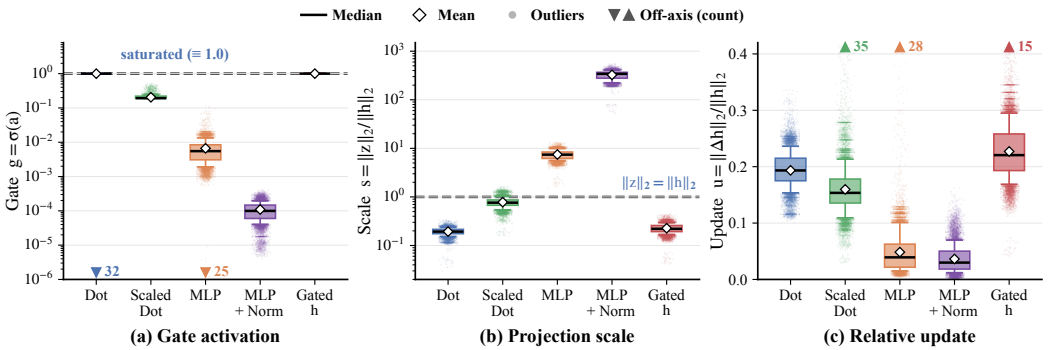


Figure 1: **The gate does not control memory injection.** Across five gate designs, (a) the gate either saturates or collapses toward zero, while (b) the projection scale shifts in the opposite direction to compensate. (c) As a result, the relative update stays within a narrow range and never vanishes.

et al., 2025). This division of labor, however, rests on a rarely examined assumption, namely that the gate actually controls the strength of the memory update.

We test this assumption on five common gate designs that share the same memory architecture, as defined in Section 3. For each design, we measure the gate value, the scale of the memory projection, and the resulting update relative to the current representation. As shown in Figure 1, the gate degenerates in two ways. The first is saturation. A dot-product gate stays near 1, and so does a gate applied to the current representation instead of the memory. The update is then set by the projection alone. The second is collapse. With an MLP gate, the values shrink toward zero while the projection grows to compensate. As a result, the gate varies widely across designs, yet the update stays within a narrow range and never vanishes. The cause is that training constrains only the final update, which is the product of the gate and the projection. Control over injection strength thus shifts to the projection, and the gate value neither reflects nor bounds how strongly memory is injected.

To restore this division of labor, we propose BIND-VLA (Bounded Injection with Normalized Direction), which decouples the direction and the magnitude of the memory update. The key idea is to assign each factor to exactly one branch. The memory projection determines only where the update points. We remove its component along the current representation and normalize the remainder, so its scale no longer matters. The gate alone determines how far the update goes, as a fraction of a fixed budget relative to the norm of the current representation. The relative update then equals the gate value times this budget, as illustrated in Figure 2(d). A larger projection can no longer compensate for a small gate. The gate thus becomes identifiable from the update, and memory injection is bounded by the budget. Orthogonalization further ties each gate value to a definite rotation of the current representation, so memory changes its direction rather than merely its length.

We embed this fusion in a memory architecture that keeps history outside the policy backbone, inspired by conditional memory (Cheng et al., 2026). As shown in Figure 2(a), past observations are stored in an external memory bank, while the current observation and instruction follow the original path of the pretrained policy. The compressor in Figure 2(b) encodes each past timestep into a single memory token. After the first backbone layer, the retriever in Figure 2(c) lets each hidden-state token read its own memory from the bank, which is then written back through the decoupled fusion.

We evaluate BIND-VLA on MIKASA-Robo (Cherepanov et al., 2026), a benchmark for memory in robotic manipulation, and on LIBERO (Liu et al., 2023) for standard manipulation. On nine Remember tasks, BIND-VLA reaches an average success rate of 38.4%, 6.0 points above dot-product gated fusion under the same architecture and 10.3 points above the same MLP gate without decoupling, so the gain comes from the fusion rule rather than the gate alone. On five tasks shared with prior work, it exceeds MemoryVLA (Shi et al., 2026b) by 11.0 points. On LIBERO, it matches the memory-free policy at 92.4% versus 92.2%, whereas dot-product fusion falls 7.4 points behind. Further analysis shows that the gate no longer degenerates and switches memory off where history is rarely needed. Our contributions are summarized as follows.

- We identify gate degeneration in memory-augmented VLAs, where the gate either saturates or collapses toward zero while the memory projection compensates, and trace it to scale coupling between the gate and the projection.

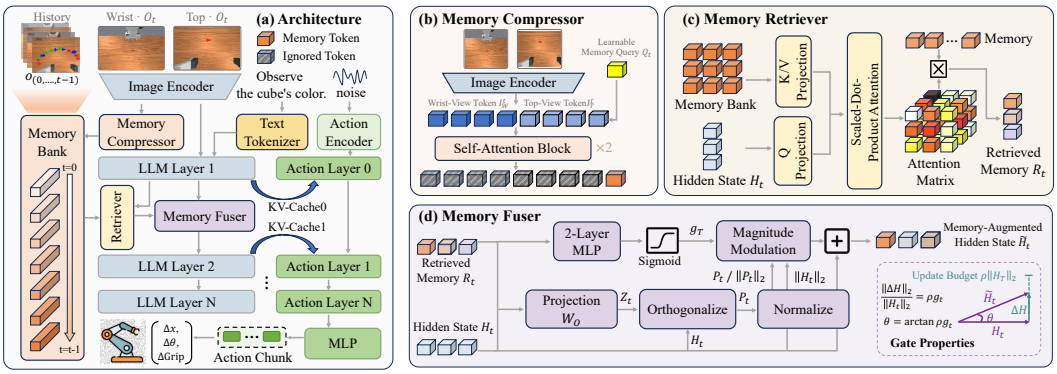


Figure 2: **Overview of BIND-VLA.** (a) History is stored in an external memory bank and fused into the backbone after its first layer. (b) The compressor encodes each past timestep into a single memory token. (c) Each hidden-state token retrieves its own memory. (d) The fuser assigns direction to the projection and magnitude to the gate, bounding the relative update by ρ.

- We propose BIND-VLA, which decouples the direction and the magnitude of the memory update, so that the gate becomes identifiable and memory injection is explicitly bounded.
- Experiments on MIKASA-Robo and LIBERO show that BIND-VLA improves memory-dependent manipulation while preserving standard performance, with a gate that adapts to the input and switches memory off when history is not needed.

2 Related Work

Memory in vision-language-action models. Most vision-language-action models build on pretrained vision-language backbones and act on the current observation alone (Zitkovich et al., 2023; Kim et al., 2025b; Ghosh et al., 2024; Black et al., 2025; Physical Intelligence et al., 2025; Qu et al., 2025; Kim et al., 2025a). Conditioning on past frames (NVIDIA et al., 2025; Li et al., 2024) adds history, but its cost grows with history length and it can induce causal confusion (De Haan et al., 2019; Wen et al., 2020; Koo et al., 2026). Other methods compress history into a few tokens (Jang et al., 2025; Koo et al., 2026; Li et al., 2026a), store it in an external memory and retrieve from it on demand (Shi et al., 2026b; Qu et al., 2026; Sridhar et al., 2026), or retrieve knowledge across episodes (Li et al., 2025; 2026b; Lin et al., 2026). These works mainly study what to store and how to retrieve it, while the gate that injects the retrieved memory is usually assumed to work as intended. We examine this assumption and design a fusion rule under which it holds.

Gating and residual update design. Multiplicative gates have long been used to control information flow (Graves, 2012; Cho et al., 2014; Srivastava et al., 2015). In pretrained models, they inject new modules without disturbing the original representation (Alayrac et al., 2022; Zhang et al., 2024; 2023; Cheng et al., 2026), and related work scales residual branches for stable training (Bachlechner et al., 2021; Touvron et al., 2021). In these designs, the gate multiplies a learnable projection, and such rescaling symmetries make individual scales non-identifiable (Dinh et al., 2017; Neyshabur et al., 2015). Normalization is a standard way to remove them (Salimans & Kingma, 2016; Van Laarhoven, 2017), and recent work further separates magnitude from direction in weights (Liu et al., 2024), in hidden states on the hypersphere (Loshchilov et al., 2025), and in residual updates orthogonal to the stream (Oh et al., 2025). BIND-VLA brings these ideas to memory fusion, letting the projection fix only the direction and a bounded gate set the magnitude.

3 Analysis of Gate Degeneration

Gated memory fusion. Memory-augmented VLAs store past observations in a memory bank M (Qu et al., 2026). Each token of the current vision-language representation h reads from the bank through attention and obtains a retrieved memory $r = \mathrm{Attn}(h, M)$. The retrieved memory is projected by $z = W_O r$ and added to h as a gated residual,

$$\widetilde{h} = h + \Delta h, \qquad \Delta h = gz, \qquad g = \sigma\big(f(h, r)\big), \tag{1}$$

where f is a scoring function and σ is the sigmoid. Retrieval decides what history to use, and the gate decides how much of it to inject. This division assumes that g controls the magnitude of Δh.

Gate designs and diagnostics. We test this assumption with five gate designs sharing the same memory architecture and training recipe. Four follow Eq. 1 and differ only in the scoring function,

$$\begin{aligned} f_{\text{Dot}} &= q^\top k/\sqrt{D}, & f_{\text{Scaled Dot}} &= q^\top k/D, \\ f_{\text{MLP}} &= \text{MLP}([h;r]), & f_{\text{MLP+Norm}} &= \text{MLP}([\mathcal{N}(h);\mathcal{N}(r)]), \end{aligned} \tag{2}$$

where $q = \text{RMSNorm}(W_q h)$, $k = \text{RMSNorm}(W_k r)$ (Zhang & Sennrich, 2019), D is the projection dimension, and $\mathcal{N}$ normalizes each input separately. Dot follows the gate of Engram (Cheng et al., 2026), and MLP follows the gated fusion of MemoryVLA (Shi et al., 2026b). Scaled Dot and MLP + Norm change only the scale of the gate input. The fifth design, Gated h, uses the Dot score but applies the gate to the current representation, giving $\widetilde{h} = gh + z$, in the spirit of gates that also weight the current state (Srivastava et al., 2015; Shi et al., 2026b). Since the update also depends on the projection, we record for every token the gate g, the projection scale $s = \|z\|_2/\|h\|_2$, and the relative update $u = \|\widetilde{h} - h\|_2/\|h\|_2$.

Observations. Figure 1 shows that the gate degenerates in two ways. The first is saturation. The Dot gate stays above 0.99 on 99.9% of tokens, and Gated h is pinned at exactly 1 on every position, reducing its update to the ungated projection z. In both designs the gate modulates nothing, and u simply follows s, which stays near 0.2. Scaled Dot avoids saturation, but its gate remains concentrated near 0.2. The second is collapse. The MLP gate drops to a mean of 0.0067 while s rises to 7.5. Normalizing its inputs pushes the gate below 5×10^{-4} on every token, while s reaches 326. Panels (a) and (b) thus mirror each other. As a result, the mean gate spans four orders of magnitude across designs, yet the mean update in panel (c) stays between 0.036 and 0.23 and never vanishes.

Why the gate degenerates. The cause lies in how the gate and the projection enter the update. With $\hat{z} = z/\|z\|_2$, Eq. 1 can be written as $\Delta h = \|h\|_2 \exp(\log g + \log s)\, \hat{z}$. The two scales act only through their sum, so any training loss $\mathcal{L}$ has identical gradients with respect to them,

$$\frac{\partial \mathcal{L}}{\partial \log g} = \frac{\partial \mathcal{L}}{\partial \log s} = \langle \nabla_{\Delta h} \mathcal{L},\, \Delta h \rangle. \tag{3}$$

Training constrains the sum, which equals $\log u$, but exerts no force on how it is split. This scale coupling agrees with Figure 1, where the product of the mean gate and the mean projection scale closely tracks the mean update in every design. The sigmoid then shapes where the split drifts. Near saturation, the gradient reaching the gate logit a is scaled by $g(1-g)$ and vanishes. The gate freezes into a constant that W_O can absorb, as happens to Dot and Gated h, whose logits grow with $\sqrt{D}$. Far below saturation, $\sigma(a) \approx e^a$. Shifting the output bias of the gate by $-\log \lambda$ divides g by λ on every input, and scaling W_O by λ restores the same update. The MLP gate can thus slide along the coupling with a single parameter, toward an ever smaller gate and an ever larger projection.

4 BIND-VLA

As shown in Figure 2(a), BIND-VLA augments a pretrained VLA policy with an external memory bank. A compressor encodes each past timestep into a single memory token. After the first backbone layer, a retriever queries the bank with the current hidden states, and a fuser injects the retrieved memory into them. The remaining layers are unchanged, and the action expert predicts an action chunk from their key-value caches. We first present the fuser, which resolves the gate degeneration of Section 3, and then the compressor and the retriever.

4.1 Direction–Magnitude Decoupled Fusion

Section 3 showed that the residual update $\Delta h = gz$ lets the projection take over the magnitude. We therefore write the update as a direction times a relative magnitude,

$$\Delta h = \alpha \|h\|_2\, d, \qquad \|d\|_2 = 1, \quad \alpha \geq 0, \tag{4}$$

and assign each factor to exactly one branch, as illustrated in Figure 2(d). Separating direction from magnitude has been used to reparameterize weights (Liu et al., 2024). Here we apply it to the update of a representation, so that the two factors can be controlled by different branches.

Table 1: **Comparison with existing VLA methods on MIKASA-Robo.** Success rates (%) over 100 episodes per task. The last four rows share the $\pi_{0.5}$ backbone and differ only in whether and how memory is injected. Best results are in bold.

Method	Shell. Touch	Intcp. Medium	Remb. Color3	Remb. Color5	Remb. Color9	Avg. Success
CronusVLA (Li et al., 2026a)	32	5	31	13	9	18.0
SpatialVLA (Qu et al., 2025)	23	27	27	17	11	21.0
OpenVLA-OFT (Kim et al., 2025a)	47	14	**59**	16	6	28.4
π_0 (Black et al., 2025)	33	42	35	22	15	29.4
MemoryVLA (Shi et al., 2026b)	88	24	44	30	20	41.2
$\pi_{0.5}$ (w/o memory)	36	**43**	20	20	10	25.8
$\pi_{0.5}$ + Dot	72	39	49	51	21	46.4
$\pi_{0.5}$ + MLP	63	36	41	48	24	42.4
BIND-VLA (Ours)	**92**	33	50	**56**	**30**	**52.2**

Direction branch. The projection $z = W_O r$ only proposes where the update should point. Normalizing z removes its norm from the update magnitude, but the update may still contain a component along h. This component only changes the length of the representation, which the normalization at the input of the next layer largely removes (Xiong et al., 2020). The effective change caused by a given gate value then depends on the alignment between z and h, rather than on the gate alone. In the extreme case where z is parallel to h, the fusion returns $(1 \pm \alpha)h$, and memory leaves the direction of h untouched. We therefore remove the parallel component before normalizing,

$$p = z - \frac{h^\top z}{\|h\|_2^2}\, h, \qquad d = \frac{p}{\|p\|_2}. \tag{5}$$

The resulting direction is orthogonal to h and invariant to the scale of z, so that each gate value corresponds to a fixed rotation of h. A similar projection was recently used in residual networks, where only the orthogonal component of each module output is added to the stream (Oh et al., 2025). Their update still scales with the module output. We additionally normalize the direction and leave its magnitude entirely to the gate.

Magnitude branch. The magnitude is set by the gate alone. A two-layer MLP computes a scalar gate from h and the retrieved memory r,

$$g = \sigma\big(W_2\, \mathrm{GELU}(W_1[h; r] + b_1) + b_2\big), \tag{6}$$

and the relative magnitude is $\alpha = \rho g$, where $\rho > 0$ is a fixed, non-learnable update budget. The gate reads r rather than z, so its value does not depend on the scale of W_O. The final fusion is

$$\widetilde{h} = h + \rho g \|h\|_2\, d. \tag{7}$$

Properties. Since $\|d\|_2 = 1$ and $h^\top d = 0$, the relative update and the rotation of h satisfy

$$u = \frac{\|\widetilde{h} - h\|_2}{\|h\|_2} = \rho g \le \rho, \qquad \angle(h, \widetilde{h}) = \arctan(\rho g). \tag{8}$$

These relations resolve both problems raised in Section 3. First, the gate becomes identifiable. Its value can be read directly from the update as $g = u/\rho$, and each value corresponds to a definite rotation of h. Second, the gate is in control. Scaling W_O by any positive constant rescales p but leaves d, g and hence Δh unchanged. The update no longer depends on the projection scale, so the coupling in Eq. 3 disappears and the projection cannot compensate for a small gate. The budget ρ further bounds how far a single injection can move the representation. Viewing the update as a bounded rotation of h relates BIND-VLA to normalized Transformers, which move hidden states on the hypersphere with controlled step sizes (Loshchilov et al., 2025).

4.2 Memory Compression and Retrieval

At a past timestep τ, a frozen visual encoder extracts features from the wrist and top views, which are concatenated into $X_\tau \in \mathbb{R}^{N \times d_v}$. As shown in Figure 2(b), the compressor appends a learnable

Table 2: **Comparison of gating and fusion designs on MIKASA-Robo.** Success rates (%) on nine Remember tasks. All configurations share the same memory architecture and differ only in how memory is gated and injected. SC$m\times n$ denotes RememberShapeAndColor with m shapes and n colors. Best results are in bold.

Configuration	Remb. Color3	Remb. Color5	Remb. Color9	Remb. Shape3	Remb. Shape5	Remb. Shape9	Remb. SC3×2	Remb. SC3×3	Remb. SC5×3	Avg. Success
No memory	20	20	10	32	12	8	15	11	11	15.4
Dot ($1/\sqrt{D}$)	49	51	21	48	31	18	36	18	**20**	32.4
Scaled Dot ($1/D$)	49	46	22	55	24	20	35	10	17	30.9
Gated h	47	52	28	**61**	32	23	34	12	16	33.9
MLP	41	48	24	56	20	12	31	11	10	28.1
MLP + Norm	16	21	10	22	15	10	16	9	8	14.1
Dot + Ortho	18	9	10	28	10	7	13	3	4	11.3
MLP + Unit	49	47	26	59	30	19	31	10	15	31.8
BIND-VLA (Ours)	**50**	**56**	**30**	58	**35**	**24**	**52**	**25**	16	**38.4**

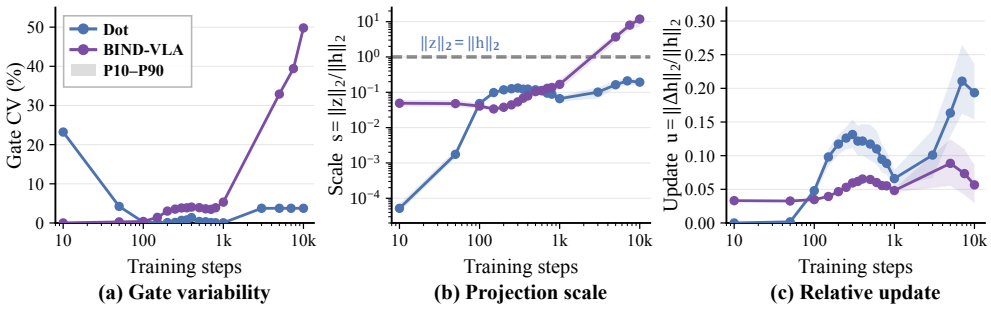


Figure 3: **Training dynamics of memory gating.** (a) The Dot gate loses its variability early, while that of BIND-VLA keeps growing. (b) The projection scale of BIND-VLA grows by over two orders of magnitude, yet (c) its update stays bounded, whereas the update of Dot follows its projection.

query q_{mem} to the projected features and processes them with a Transformer encoder E_θ. Only the output at the query position is kept as the memory token,

$$m_\tau = \big[E_\theta([P(X_\tau); q_{\text{mem}}])\big]_{q_{\text{mem}}}, \qquad M_t = [m_0; \ldots; m_{t-1}]. \tag{9}$$

Each timestep thus contributes one token, and a history of length T occupies T memory tokens instead of TN visual tokens. A single token is a tight bottleneck, and the action loss alone gives little guidance on what it should keep. We therefore pretrain the compressor with a reconstruction objective. A causal decoder D_ϕ takes m_τ as its first token and, under teacher forcing, predicts each feature token from the memory token and the preceding ground-truth tokens,

$$\widehat{X}_{\tau,j} = D_\phi(m_\tau, X_{\tau,<j}), \qquad \mathcal{L}_{\text{rec}} = \frac{1}{N d_v} \sum_{j=1}^{N} \big\| \widehat{X}_{\tau,j} - X_{\tau,j} \big\|_2^2. \tag{10}$$

The memory token is the only path from the encoder to the decoder, so minimizing $\mathcal{L}_{\text{rec}}$ encourages it to summarize both views. The decoder is used only during pretraining and is discarded afterwards.

Contextual retrieval. Let $H_t \in \mathbb{R}^{L\times d}$ denote the hidden states after the first backbone layer. As shown in Figure 2(c), the retriever performs multi-head cross-attention with H_t as queries and M_t as keys and values,

$$R_t = \text{MHA}(H_t, M_t, M_t). \tag{11}$$

Every token retrieves on its own, so each hidden state h obtains its own readout r. History is stored per timestep, while its selection depends on the current token. Each pair (h, r) is then fused by Eq. 7 before entering the next backbone layer. Because the memory tokens serve only as keys and values, they never occupy positions in the backbone, whose sequence length remains L throughout.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate BIND-VLA on MIKASA-Robo (Cherepanov et al., 2026) for memory-dependent manipulation and on LIBERO (Liu et al., 2023) for standard manipulation. From MIKASA-Robo, we use nine Remember tasks covering Remember Color, Remember Shape and Remember Shape–Color. In each task, the target is shown only at the beginning of the episode, and the policy must later select it among several candidates once the cue is no longer visible. To compare with prior work, we also report ShellGameTouch and InterceptMedium. From LIBERO, we use the Spatial, Object, Goal and LIBERO-10 suites, which rarely require history and thus test whether attaching memory harms a policy that could act on the current observation alone.

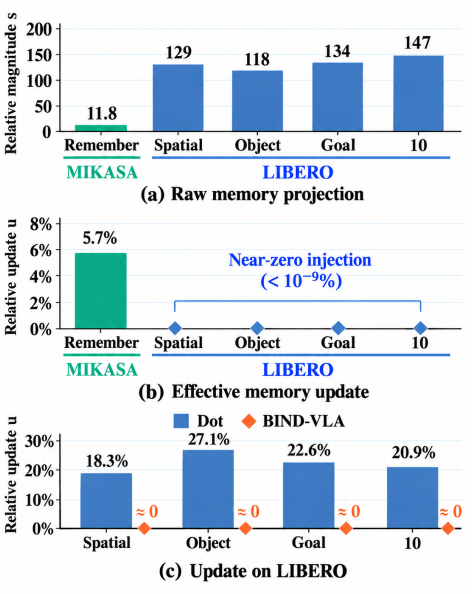


Figure 4: **Memory injection across MIKASA and LIBERO.**

Implementation details. All models build on the pretrained $\pi_{0.5}$ policy (Physical Intelligence et al., 2025) with a frozen visual encoder. The memory-augmented models are trained in two stages, each for 10k steps on 24 NVIDIA A800 GPUs. We first pretrain the memory compressor with $\mathcal{L}_{\text{rec}}$ on the training demonstrations, and then optimize the policy and the full memory module jointly with the action objective of $\pi_{0.5}$. Memory is injected after the first backbone layer, with $\rho = 0.7$ unless otherwise stated. We run 100 evaluation episodes per task on MIKASA-Robo and 50 on LIBERO, and clear the memory bank at the start of each episode. Further details are in Appendix B.

5.2 MAIN RESULTS

Comparison with existing methods. Table 1 compares BIND-VLA with existing VLA methods on five MIKASA-Robo tasks. BIND-VLA reaches an average success rate of 52.2%, 11.0 points above MemoryVLA, the strongest prior method. The margins are largest on RememberColor5 and RememberColor9, where BIND-VLA exceeds MemoryVLA by 26 and 10 points and reaches 56% and 30% against chance levels of 20% and 11%, while π_0 stays close to chance on all three RememberColor tasks. Since these methods differ in backbone and pretraining, the lower block of Table 1 further compares memory designs on the same $\pi_{0.5}$ backbone. The memory-free $\pi_{0.5}$ reaches 25.8%, well below MemoryVLA, so the stronger backbone alone does not account for the gain. On this backbone, Dot and a MemoryVLA-style MLP gate reach 46.4% and 42.4%, the latter close to MemoryVLA itself, and BIND-VLA improves on them by 5.8 and 9.8 points. The advantage of BIND-VLA therefore stems mainly from its fusion rule rather than from the backbone.

Comparison of gating and fusion designs. Table 2 compares fusion designs within the same memory architecture. Adding memory with Dot raises the average from 15.4% to 32.4%, confirming that the retrieved history is useful. The other gate designs bring no further gain, following the degeneration identified in Section 3. Scaled Dot and Gated h stay within 1.5 points of Dot, while the collapsed MLP and MLP + Norm fall to 28.1% and 14.1%, the latter below the memory-free policy. Changing the gate parameterization alone is therefore not enough, and neither is decoupling alone. Dot + Ortho applies our full decoupled fusion to the Dot gate and reaches only 11.3%, as its gate saturates at 0.99 and pins the relative update near the full budget on almost every token. BIND-VLA combines decoupled fusion with an MLP gate and reaches 38.4%, 6.0 points above Dot and 10.3 points above MLP with the same gate, and is best on seven of the nine tasks. Both steps of the decoupling are needed. MLP + Unit normalizes the projection without orthogonalizing it. Its gate is already identifiable, since $u = \rho g$ still holds, yet it reaches only 31.8%, 6.6 points

Table 3: **Comparison on LIBERO.** Success rates (%) on four suites. The last three rows share the $\pi_{0.5}$ backbone and differ only in whether and how memory is injected. Best results are in bold.

Method	Spatial	Object	Goal	LIBERO-10	Avg. Success
MaIL (Jia et al., 2025)	74.3	90.1	81.8	78.6	81.2
SpatialVLA (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.0
TraceVLA (Zheng et al., 2025)	84.6	85.2	75.1	54.1	74.8
OpenVLA (Kim et al., 2025b)	84.7	88.4	79.2	53.7	76.5
CoT-VLA (Zhao et al., 2025)	87.5	91.6	87.6	69.0	83.9
CronusVLA (Li et al., 2026a)	90.1	**94.7**	91.3	68.7	86.2
TriVLA (Liu et al., 2025)	91.2	93.8	89.8	73.2	87.0
$\pi_{0.5}$ (w/o memory) (Physical Intelligence et al., 2025)	**91.8**	92.0	93.2	91.8	92.2
$\pi_{0.5}$ + Dot	87.8	86.8	81.0	83.6	84.8
BIND-VLA (Ours)	91.6	92.4	**93.4**	**92.2**	**92.4**

Table 4: **Ablations of BIND-VLA.** Average success rates (%) on nine Remember tasks. (a) Compression and retrieval, with injection at layer 1. (b) Injection layer, with encoder compression and per-token retrieval. (c) Update budget ρ. Shaded rows denote the default configuration.

(a) Compression and retrieval

Compression	Retrieval	Avg.
Mean pooling	Pooled	23.4
Encoder	Pooled	31.2
Encoder	Per-token	**38.4**

(b) Injection layer

Layer	Avg.
1	**38.4**
5	33.6
11	25.7

(c) Update budget

ρ	Avg.
0.10	31.6
0.70	**38.4**
0.99	26.7

below BIND-VLA, with the largest gaps on SC3×2 and SC3×3. Without orthogonalization, part of each update only rescales h, which the normalization of the next layer largely removes, so the same gate value no longer yields a fixed change of the representation. The gate and the fusion rule must therefore be designed together. Decoupling helps only when the gate can still learn, and a learnable gate controls injection only when decoupling prevents the projection from compensating.

Performance on standard manipulation. Table 3 tests whether attaching memory harms tasks that rarely require history. BIND-VLA reaches an average of 92.4%, on par with the memory-free $\pi_{0.5}$ policy at 92.2%, and stays within 0.4 points of it on every suite. Attaching memory through Dot instead lowers the average to 84.8%, a drop of 7.4 points that appears on all four suites and is largest on Goal, where success falls from 93.2% to 81.0%. The two fusions differ in whether memory can be switched off. The Dot gate saturates near 1 regardless of the input (Section 3), so retrieved history is injected into every step of a policy that does not need it.

5.3 Analysis of Memory Gating

Training dynamics. Figure 3 tracks the gate variability, measured as the coefficient of variation of g across tokens, together with the projection scale and the relative update. The Dot gate saturates early, with its mean reaching 0.998 after about 100 steps and its variability staying below 5% thereafter, so its update simply follows its projection scale. The gate variability of BIND-VLA instead keeps growing after about 1k steps and reaches about 50% at 10k steps. Its projection scale rises by more than two orders of magnitude, from 3.69 to 11.83 between 5k and 10k steps alone, yet its update falls from 0.089 to 0.057 over the same interval, tracking the gate mean as $u = \rho g$ predicts. The projection grows freely but no longer takes over injection strength.

Gating across benchmarks. We apply the same probing protocol to BIND-VLA trained on MIKASA-Robo and on LIBERO, and to Dot trained on LIBERO. On the Remember tasks, the gate of BIND-VLA remains active, with a mean of 0.081 and a relative update of about 0.057 (Figure 4(b)). On LIBERO, its gate mean drops to between 10^{-14} and 10^{-12}, and injection is effectively switched off. Its projection scale meanwhile reaches 118 to 147, an order of magnitude above the 11.8 on the Remember tasks (Figure 4(a)). Under residual fusion, such growth would restore a non-zero update, as it does for MLP + Norm in Section 3. Under decoupled fusion it has no effect, so unlike the collapsed MLP gates, the update vanishes with the gate, and a small gate means that

memory is switched off rather than compensated by the projection. Dot instead leaves the update to the projection through its saturated gate (Figure 4(c)). Its relative update remains between 0.18 and 0.27, comparable to about 0.2 on the Remember tasks (Figure 1), so memory is injected at a similar strength whether or not the task requires history. This explains why BIND-VLA preserves the performance of the memory-free policy on LIBERO while Dot loses 7.4 points.

Inference-time interventions. To examine how each model uses memory, we modify the memory update at inference time without retraining (Table 5). *Closed* sets $\Delta h = 0$ and removes memory entirely. *Flattened* replaces each update by $\bar{u}\,\|h\|_2\,\Delta h/\|\Delta h\|_2$, where $\bar{u}$ is the mean relative update of the model. It preserves the direction of each update and the average injection strength, but removes the dependence of the magnitude on the input, and for BIND-VLA it is equivalent to fixing $g = \bar{g}$. *Saturated* sets $g = 1$ in BIND-VLA, so that $u = \rho$ on every token. Closing the update lowers Dot, MLP and BIND-VLA by 21.5, 16.7 and 25.7 points, so all three models rely on the retrieved history. Flattening changes Dot by only 1.8 points, within evaluation noise and consistent with its update being set by the projection rather than the gate. It lowers MLP by 5.4 points and BIND-VLA by 9.2 points, the latter on all nine tasks. The benefit of input-dependent modulation thus grows with the degree to which the gate controls the update. Saturation fixes the relative update at $\rho = 0.7$, an order of magnitude above the learned 0.057, and reduces the average to 1.0%, so stronger injection does not help.

Table 5: **Inference-time interventions.** Average success rates (%) on nine Remember tasks without retraining.

Model	Intervention	Avg.	Δ
Dot	Learned	32.4	–
	Closed	10.9	−21.5
	Flattened	30.6	−1.8
MLP	Learned	28.1	–
	Closed	11.4	−16.7
	Flattened	22.7	−5.4
BIND-VLA	Learned	**38.4**	–
	Closed	12.7	−25.7
	Flattened	29.2	−9.2
	Saturated	1.0	−37.4

5.4 Ablation Studies

Architectural ablations. We ablate the compression method, the retrieval granularity and the injection depth in Table 4(a) and (b), keeping all other components at the default configuration. Replacing mean pooling with the learned encoder raises the average success rate from 23.4% to 31.2%, as the encoder is trained to reconstruct both views and retains more information than a plain average. Per-token retrieval raises it further to 38.4%. The two retrieval variants share the same decoupled fusion and gate, and differ only in whether the query is a pooled summary of the hidden states or each hidden state itself. The gain of 7.2 points therefore reflects the retrieval granularity, consistent with each token reading the history relevant to it rather than a shared summary. Moving injection from layer 1 to layers 5 and 11 lowers success to 33.6% and 25.7%. A possible explanation is that memory compressed from visual features is closer to early-layer representations, and that early injection leaves more depth to integrate history into the decision.

Sensitivity to the update budget. We retrain BIND-VLA with different update budgets ρ. As shown in Table 4(c), the average success rate is 31.6% at $\rho = 0.1$, peaks at 38.4% at $\rho = 0.7$, and drops to 26.7% at $\rho = 0.99$. A small budget caps the largest update memory can apply, which limits tasks that need strong recall. A large budget makes the update more sensitive to the gate, since the same change in g moves the representation by ρ times as much. Since ρ only sets an upper bound, the actual injection strength ρg is still learned by the gate.

6 Conclusion

In this work, we identified gate degeneration in memory-augmented VLAs. Across common designs, the gate saturates or collapses, because scale coupling lets the projection take over injection strength. BIND-VLA decouples the direction and magnitude of the memory update, making the gate identifiable and injection bounded. It improves average success on MIKASA-Robo by 6.0 points over dot-product gated fusion, while on LIBERO its gate switches memory off and matches the memory-free policy. More broadly, a gate controls a residual branch only if the branch cannot rescale its update, which may hold for other gated modules in pretrained models. Future work will extend BIND-VLA to real robots and longer histories.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/bachlechner21a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action flow model for general robot control. In *Robotics: Science and Systems*, 2025.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Han Zhang, Yu-Kun Li, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4968–4990, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.226. URL https://aclanthology.org/2026.acl-long.226/.

Egor Cherepanov, Nikita Kachaev, Alexey Kovalev, and Aleksandr Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *International Conference on Learning Representations*, 2026.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1019–1028. PMLR, 2017.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090.

Alex Graves. Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*, pp. 37–45. Springer, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-24797-2_4.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-Language-Action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

Xiaogang Jia, Qian Wang, Atalay Donat, Bowen Xing, Ge Li, Hongyi Zhou, Onur Celik, Denis Blessing, Rudolf Lioutikov, and Gerhard Neumann. MaIL: Improving imitation learning with selective state space models. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3888–3907. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/jia25b.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source Vision-Language-Action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025b.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your Vision-Language-Action model into a history-aware policy. In *International Conference on Learning Representations*, 2026.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, and Jiangmiao Pang. CronusVLA: Towards efficient and robust manipulation via multi-frame vision-language-action modeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18388–18396, 2026a. doi: 10.1609/aaai.v40i22.38903. URL https://ojs.aaai.org/index.php/AAAI/article/view/38903.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. MAP-VLA: Memory-augmented prompting for Vision-Language-Action model in robotic manipulation. *arXiv preprint arXiv:2511.09516*, 2025.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024.

Zaijing Li, Bing Hu, Rui Shao, Gongwei Chen, Dongmei Jiang, Pengwei Xie, Jianye Hao, and Liqiang Nie. Global prior meets local consistency: Dual-memory augmented vision-language-action model for efficient robotic manipulation. *arXiv preprint arXiv:2602.20200*, 2026b.

Min Lin, Xiwen Liang, Bingqian Lin, Jingzhi Liu, Zijian Jiao, Kehan Li, Ziang Yan, Yu Sun, Weijia Liufu, Yuhan Ma, Jiarui Hu, Yuecheng Liu, Shen Zhao, Yuzheng Zhuang, and Xiaodan Liang. EchoVLA: Robotic Vision-Language-Action model with synergistic declarative memory for mobile manipulation. In *European Conference on Computer Vision*, 2026. URL https://arxiv.org/abs/2511.18112.

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

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 21–27 Jul 2024.

Zhenyang Liu, Yongchong Gu, Sixiao Zheng, Yanwei Fu, Xiangyang Xue, and Yu-Gang Jiang. TriVLA: A triple-system-based unified Vision-Language-Action model with episodic world modeling for general robot control. *arXiv preprint arXiv:2507.01424*, 2025.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *International Conference on Learning Representations*, 2025.

Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-SGD: Path-normalized optimization in deep neural networks. *Advances in Neural Information Processing Systems*, 28, 2015.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Giyeong Oh, Woohyun Cho, Siyeol Kim, Suhwan Choi, and Youngjae Yu. Revisiting residual connections: Orthogonal updates for stable and efficient deep networks. *Advances in Neural Information Processing Systems*, 38:71725–71759, 2025. doi: 10.52202/085713-2409.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A Vision-Language-Action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for Visual-Language-Action models. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.011.

Hongyu Qu, Jianzhe Gao, Xiaobin Hu, Shaohuan Yang, Xinlei Yu, Rui Yan, Wenguan Wang, Xiangbo Shu, and Shuicheng Yan. Dual latent memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2607.07608*, 2026.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. MemoryVLA++: Temporal modeling via memory and imagination in Vision-Language-Action models. *arXiv preprint arXiv:2606.09827*, 2026a.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in Vision-Language-Action models for robotic manipulation. In *International Conference on Learning Representations*, 2026b.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 97142–97166, 2026.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in Neural Information Processing Systems*, 28, 2015.

Hao Sun, Yu Song, Shiyu Teng, Ziwei Niu, and Yen-Wei Chen. MIRTH: Mutual-information reasoning with temporal hubs for vision-language-action agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22199–22215, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1016. URL https://aclanthology.org/2026.acl-long.1016/.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42. IEEE, 2021.

Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

Ziye Wang, Modi Shi, Chaojun Ni, Jiazhi Yang, Mengdi Li, Zhizhong Su, Tianwei Lin, and Hongyang Li. NativeMEM: Native memory compression for long-horizon robotic manipulation. *arXiv preprint arXiv:2607.06678*, 2026.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. *Advances in Neural Information Processing Systems*, 33:2564–2575, 2020.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3813–3824. IEEE, 2023.

Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. LLaMA-Adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=d4UiXAHN2W`.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. CoT-VLA: Visual chain-of-thought reasoning for Vision-Language-Action models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713. IEEE, 2025.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-Language-Action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023.

APPENDIX

A LLM Usage **15**

B Implementation Details **15**

B.1 Training Protocol 15

B.2 Architecture 15

B.3 Initialization 16

B.4 Numerical Stability 16

B.5 Parameters and Computational Cost 16

B.6 Inference 17

C Gate and Fusion Designs **17**

C.1 Shared Components 17

C.2 Gate Functions 17

C.3 Fusion Rules 18

C.4 Baselines 18

D Theoretical Details **19**

D.1 Scale Coupling in Gated Residual Fusion 19

D.2 Saturation and Collapse of the Sigmoid Gate 20

D.3 Properties of Decoupled Fusion 20

D.4 Scale Invariance and Gradient Decomposition 21

E Diagnostic Protocol **22**

E.1 Probing Protocol 22

E.2 Gate Designs (Figure 1) 22

E.3 Training Dynamics (Figure 3) 23

E.4 Memory Injection across Benchmarks (Figure 4) 23

E.5 Coefficient of Variation 23

F Benchmark Details **23**

F.1 MIKASA-Robo 23

F.2 LIBERO 25

G Visualization of Memory Gating **26**

H Additional Ablations **28**

H.1 Injection Layer 28

H.2 Update Budget 28

I Qualitative Examples **28**

A LLM USAGE

In preparing this manuscript, we used Large Language Models (LLMs) exclusively as writing aids for improving grammar, clarity, and readability. All LLM-suggested edits were carefully reviewed and revised by the authors. The research questions, methods, experimental design, results, and conclusions were conceived and produced entirely by the authors, who bear full responsibility for the content of this paper.

B IMPLEMENTATION DETAILS

This section describes the training protocol, architecture, initialization and computational cost of BIND-VLA. Unless otherwise stated, all numbers refer to the default configuration used in the main experiments, namely encoder compression, per-token retrieval, injection after the first backbone layer, an MLP gate and decoupled fusion with $\rho = 0.7$. All gate and fusion designs compared in the main text follow the same training protocol.

B.1 TRAINING PROTOCOL

Training proceeds in two stages, each for 10k steps on 24 NVIDIA A800 GPUs. In the first stage, we pretrain the memory compressor with the reconstruction objective $\mathcal{L}_{\text{rec}}$ in Eq. 10, while the policy is kept frozen. Only the compressor encoder E_θ, the decoder D_ϕ and the memory query q_{mem} are updated. In the second stage, we discard $\mathcal{L}_{\text{rec}}$ and optimize the policy together with the compressor encoder, the retriever and the gate under the action objective of $\pi_{0.5}$. The decoder is discarded after the first stage and is not used in the second stage or at inference. The visual encoder stays frozen throughout both stages. Table 6 lists the hyperparameters.

The learning rate follows a linear warmup and a cosine decay whose horizon far exceeds the number of training steps, so it stays effectively constant after warmup. MIKASA-Robo actions are 7-dimensional and are zero-padded to the 32-dimensional action space of $\pi_{0.5}$.

Table 6: **Training hyperparameters.** Batch sizes are global.

Hyperparameter	Stage 1 (compressor)	Stage 2 (joint)
GPUs	$24 \times$ NVIDIA A800	
Training steps	10,000	
Global batch size	96	48
Warmup steps	3,333	
Peak learning rate	8.66×10^{-5}	
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$)	
Weight decay	10^{-10}	
Gradient clipping	global ℓ_2 norm 1.0	
Learning rate schedule	linear warmup, cosine decay	
Precision	bfloat16	
Action chunk length	10	
Random seed	42	
Loss	$\mathcal{L}_{\text{rec}}$	action loss of $\pi_{0.5}$
Trainable modules	E_θ, D_ϕ, q_{mem}	policy (w/o visual encoder), E_θ, q_{mem}, retriever, gate

B.2 ARCHITECTURE

Visual tokens. Each camera image is resized to 224×224 and encoded by the frozen SigLIP So400m encoder of $\pi_{0.5}$ with patch size 14, giving 256 tokens per view. The top and wrist views are concatenated along the token dimension, so each timestep yields $N = 512$ tokens of dimension $d_v = 1152$. The projection P in Eq. 9 is the pretrained multimodal projector of $\pi_{0.5}$, a single linear layer that maps the features to the backbone width $d = 2048$. The compressor, the retriever and the backbone therefore share the same width.

Compressor. The encoder E_θ is a two-layer pre-norm Transformer encoder with width 2048, 8 attention heads, a feed-forward dimension of 8192, ReLU activation and dropout 0.1, without a final

normalization layer. The learnable memory query $q_{\text{mem}} \in \mathbb{R}^{2048}$ is appended after the N visual tokens, and its output is kept as the memory token m_τ. The decoder D_ϕ has the same architecture with a causal attention mask and a linear output head. Following Eq. 10, it takes m_τ as its first token and, under teacher forcing, predicts each feature token from the preceding ones. The reconstruction targets are computed with gradients stopped, and the loss is averaged over tokens and feature dimensions.

Retriever. The retriever is a single multi-head cross-attention layer with 8 heads of dimension 256. The query, key, value and output projections are linear layers of size 2048×2048 with bias, and attention logits are scaled by $1/\sqrt{256}$. Every hidden-state token after the first backbone layer acts as a query, and the memory tokens M_t act as keys and values.

Gate. The gate in Eq. 6 takes the concatenation $[h; r] \in \mathbb{R}^{4096}$ without input normalization, and maps it through a hidden layer of width 256 with GELU activation to a scalar logit. Each token thus receives its own scalar gate g, shared across all feature dimensions.

Fusion. In Eqs. 5 and 7, we treat h as a reference frame and stop gradients through $h/\|h\|_2$ and $\|h\|_2$. Gradients still reach h through the residual path of Eq. 7. The update budget is $\rho = 0.7$.

History. The memory bank stores one memory token for every past timestep of the current episode, without truncation. Remember tasks have about 15 timesteps of history on average. At the first timestep the bank is empty and the fusion is skipped, so $\widetilde{h} = h$. History images are not augmented. For long histories, the visual encoder and the compressor process past frames in chunks of 32.

B.3 Initialization

The fusion is initialized close to the identity. For the gate, W_1 is drawn from a truncated normal distribution with standard deviation 0.02 and $b_1 = 0$, while $W_2 = 0$ and $b_2 = -3$. At initialization, the gate is therefore constant across inputs, $g_0 = \sigma(-3) \approx 0.047$, and the relative update is $\rho g_0 \approx 0.033$. Setting $W_2 = 0$ lets the gate start from a constant rather than from a random function of (h, r), and its input dependence is learned during training.

The output projection W_O cannot be initialized to zero under decoupled fusion, since the direction d in Eq. 5 is undefined at $z = 0$. We initialize W_O from a truncated normal distribution with standard deviation 0.02 and zero bias, and rely on the small initial gate to keep the update small. The retriever projections W_Q, W_K and W_V use the same initialization. For the residual fusion designs of Section 3, we instead set $W_O = 0$, which makes the fusion exactly the identity at initialization. The initialization of the other gate designs is given in Appendix C.

B.4 Numerical Stability

We clamp all norms in Eqs. 5 and 7 from below by $\epsilon = 10^{-6}$. When z is nearly parallel to h, the orthogonal component p becomes small and $d = p/\max(\|p\|_2, \epsilon)$ shrinks with it. In the degenerate case $p = 0$, the update vanishes, $\Delta h = 0$, instead of producing a division by zero. We did not observe this case in practice. The projection scale $s = \|z\|_2/\|h\|_2$ reported in Section 3 is measured before normalization, while the relative update of BIND-VLA equals ρg regardless of s.

B.5 Parameters and Computational Cost

Table 7 breaks down the parameters of BIND-VLA. The memory module used at inference, consisting of the compressor encoder, the retriever and the gate, adds 118.6M parameters, 3.3% of the $\pi_{0.5}$ backbone. The decoder adds another 104.9M parameters, but is used only in the first stage.

The second stage takes about 1.4 s per step on 24 GPUs, or about 4 hours for 10k steps. We use gradient checkpointing for the backbone, the action expert and the compressor. Since each past timestep is compressed into a single token, the additional cost per step consists of encoding the new frame once and one cross-attention over T memory tokens per hidden-state token, which is small compared with the backbone for the history lengths considered here.

Table 7: **Parameter count of BIND-VLA.**

Module	Parameters	Used in
$\pi_{0.5}$ backbone	3,616.8M	Stages 1, 2, inference
Compressor encoder E_θ and q_{mem}	100.7M	Stages 1, 2, inference
Compressor decoder D_ϕ	104.9M	Stage 1
Retriever	16.8M	Stage 2, inference
Gate MLP	1.0M	Stage 2, inference
Memory module at inference	118.6M	

B.6 Inference

Actions are sampled with 10 Euler steps of the flow-matching action expert of $\pi_{0.5}$. The policy predicts a chunk of 10 actions and executes the first 5 before replanning. The memory bank is updated at every environment step rather than only at replanning. After each step, the new observation is compressed into a memory token and appended to the bank, so that every executed timestep contributes one entry, matching the history seen during training. The bank is cleared at the start of each episode, and dropout in the compressor is disabled at inference.

C Gate and Fusion Designs

This section gives the complete formulation of every configuration compared in Section 3 and Table 2, and describes how the baselines in Tables 1 and 3 are obtained.

C.1 Shared Components

All configurations share the compressor, the retriever, the injection layer and the training protocol of Appendix B. They differ only in how the gate g is computed and how the retrieved memory is written into the hidden state. For a hidden-state token $h \in \mathbb{R}^D$ with $D = 2048$, the retriever computes

$$r = \text{Concat}_{j=1}^{8}\left(\text{softmax}\left(q^{(j)\top}K^{(j)}/\sqrt{d_h}\right)V^{(j)}\right), \qquad z = W_O r, \tag{12}$$

where $q = W_Q h$, $K = W_K M_t$, $V = W_V M_t$, the superscript (j) denotes the j-th head and $d_h = 256$. The attention temperature is the same in all configurations and independent of the gate. When the memory bank is empty, the update is skipped and $\widetilde{h} = h$.

C.2 Gate Functions

Each gate outputs a scalar $g = \sigma(a) \in (0, 1)$ per token from a logit a.

Dot. Following Engram (Cheng et al., 2026), the logit is a scaled dot product between normalized projections of h and r,

$$a = \frac{q_g^\top k_g}{\sqrt{D}}, \qquad q_g = \text{RMSNorm}(W_{gq}h), \qquad k_g = \text{RMSNorm}(W_{gk}r), \tag{13}$$

where $W_{gq}, W_{gk} \in \mathbb{R}^{D\times D}$ are initialized from a truncated normal distribution with standard deviation 0.02, and both RMSNorm layers have learnable scales. At initialization the dot product is close to zero, so $g_0 \approx 0.5$.

Scaled Dot. Identical to Dot except that the logit is divided by D instead of $\sqrt{D}$, $a = q_g^\top k_g / D$. This reduces the scale of the logit by a factor of $\sqrt{D} \approx 45$ and tests whether saturation can be avoided by rescaling alone.

MLP. Following the gated fusion of MemoryVLA (Shi et al., 2026b), the logit is computed by a two-layer MLP on the concatenated inputs,

$$a = W_2\ \text{GELU}\left(W_1[h; r] + b_1\right) + b_2, \tag{14}$$

with a hidden width of 256. W_1 is initialized from a truncated normal distribution with standard deviation 0.02, $b_1 = 0$ and $W_2 = 0$, so the gate is constant at initialization, $g_0 = \sigma(b_2)$.

MLP + Norm. Identical to MLP except that both inputs are normalized separately by RMSNorm without learnable scales before concatenation, $a = \mathrm{MLP}([\mathcal{N}(h); \mathcal{N}(r)])$. This normalizes the gate input, not the projection z.

C.3 Fusion Rules

Residual. The gated residual of Eq. 1,

$$\widetilde{h} = h + gz. \tag{15}$$

We initialize $W_O = 0$, so that the fusion is exactly the identity at initialization. The relative update $u = g\|z\|_2/\|h\|_2$ depends on the scale of W_O.

Gated h. The gate is applied to the current representation instead of the memory,

$$\widetilde{h} = gh + z, \tag{16}$$

with $W_O = 0$ at initialization. When the memory bank is empty, we set $g = 1$, which again gives $\widetilde{h} = h$.

Unit. The projection is normalized but not orthogonalized,

$$\widetilde{h} = h + \rho g \|h\|_2 \, \hat{z}, \qquad \hat{z} = \frac{z}{\|z\|_2}. \tag{17}$$

The relative update equals ρg and no longer depends on the scale of W_O, but $\hat{z}$ may retain a component along h, which only rescales the representation.

Ortho. The decoupled fusion of Eqs. 5 and 7, which removes the component of z along h before normalization,

$$\widetilde{h} = h + \rho g \|h\|_2 \, d, \qquad d = \frac{p}{\|p\|_2}, \qquad p = z - \frac{h^\top z}{\|h\|_2^2} \, h. \tag{18}$$

For both Unit and Ortho, the direction is undefined at $z = 0$, so W_O is initialized from a truncated normal distribution with standard deviation 0.02 instead of zero. Near-identity initialization is instead obtained through a small initial gate, as described below. The stop-gradient and numerical treatment of Appendix B.4 apply to both.

Configurations Table 8 lists all configurations as combinations of a gate function and a fusion rule. The residual designs reach the identity at initialization through $W_O = 0$, and therefore leave the gate bias at its default $b_2 = 0$. The decoupled designs with an MLP gate cannot use $W_O = 0$, and instead set $b_2 = -3$, so that the initial relative update is $\rho\sigma(-3) \approx 0.033$. The Dot gate has no output bias, and Dot + Ortho therefore starts from a relative update of about $\rho/2 = 0.35$.

The configurations are chosen so that pairs of rows isolate individual factors. Dot versus Scaled Dot and MLP versus MLP + Norm change only the scale of the gate input. Dot versus Gated h changes only where the gate is applied. Dot versus Dot + Ortho and MLP versus BIND-VLA change only the fusion rule for a fixed gate. MLP + Unit versus BIND-VLA changes only whether the projection is orthogonalized. The No memory configuration removes the compressor, the retriever and the fusion, and fine-tunes the policy with the same protocol.

C.4 Baselines

MIKASA-Robo. In Table 1, the three $\pi_{0.5}$-based memory variants and the memory-free $\pi_{0.5}$ are trained and evaluated by us under the same protocol as BIND-VLA. Since neither the $\pi_{0.5}$ release nor MIKASA-Robo reports results for $\pi_{0.5}$ on this benchmark, we fine-tune the official $\pi_{0.5}$ base checkpoint on the same demonstrations with the second-stage recipe of Appendix B.1, without the memory module. $\pi_{0.5}$ + Dot and $\pi_{0.5}$ + MLP correspond to the Dot and MLP configurations above. All four are evaluated with the same 100 episodes per task and the same evaluation seeds. Results for the remaining methods are taken from Shi et al. (2026b).

Table 8: **Summary of gate and fusion configurations.** q_g and k_g are the normalized gate projections of h and r, $\hat{z} = z/\|z\|_2$, and d is the orthogonalized direction of Eq. 5. TN denotes a truncated normal initialization with standard deviation 0.02, and g_0 is the gate value at initialization.

Configuration	Gate logit a	Fusion $\widetilde{h}$	ρ	b_2	W_O init	g_0
Dot	$q_g^\top k_g/\sqrt{D}$	$h + gz$	–	–	0	≈ 0.5
Scaled Dot	$q_g^\top k_g/D$	$h + gz$	–	–	0	≈ 0.5
Gated h	$q_g^\top k_g/\sqrt{D}$	$gh + z$	–	–	0	≈ 0.5
MLP	$\mathrm{MLP}([h; r])$	$h + gz$	–	0	0	0.5
MLP + Norm	$\mathrm{MLP}([\mathcal{N}(h); \mathcal{N}(r)])$	$h + gz$	–	0	0	0.5
Dot + Ortho	$q_g^\top k_g/\sqrt{D}$	$h + \rho g\|h\|_2\, d$	0.7	–	TN	≈ 0.5
MLP + Unit	$\mathrm{MLP}([h; r])$	$h + \rho g\|h\|_2\, \hat{z}$	0.7	-3	TN	0.047
BIND-VLA	$\mathrm{MLP}([h; r])$	$h + \rho g\|h\|_2\, d$	0.7	-3	TN	0.047

LIBERO. In Table 3, $\pi_{0.5}$ (w/o memory), $\pi_{0.5}$ + Dot and BIND-VLA are fine-tuned by us from the $\pi_{0.5}$ base checkpoint on each suite with the second-stage recipe of Appendix B.1, and evaluated with 50 episodes per task. Our memory-free $\pi_{0.5}$ is lower than the officially released LIBERO checkpoint, which is trained with a larger batch size, a longer schedule and an exponential moving average of the weights. We retrain all three under the same budget so that they differ only in the memory module. Results for the remaining methods are taken from their original papers.

D Theoretical Details

This section provides the derivations behind Sections 3 and 4. We first formalize the scale coupling of gated residual fusion and characterize how the sigmoid determines where the gate drifts. We then prove the properties of decoupled fusion, show that it removes the coupling at the level of gradients, and explain why orthogonalization is needed in addition to normalization. Throughout, $h \in \mathbb{R}^D$ is a hidden-state token, r its retrieved memory, $z = W_O r$ the memory projection, $\mathcal{L}$ any differentiable training loss and $\nabla\mathcal{L} = \nabla_{\Delta h}\mathcal{L}$ its gradient with respect to the update. In all gate designs of Appendix C, the gate reads (h, r) and does not depend on W_O.

D.1 Scale Coupling in Gated Residual Fusion

For $z \neq 0$, write $\hat{z} = z/\|z\|_2$ and $s = \|z\|_2/\|h\|_2$. The gated residual update of Eq. 1 factorizes into a direction and two scalar scales,

$$\Delta h = gz = \|h\|_2 \exp(\alpha + \beta)\, \hat{z}, \qquad \alpha = \log g, \qquad \beta = \log s. \tag{19}$$

Proposition 1 (Scale coupling)**.** *For the update in Eq. 19, the following hold.*

(i) The loss has identical sensitivity to the two scales,

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \frac{\partial \mathcal{L}}{\partial \beta} = \langle \nabla\mathcal{L}, \Delta h \rangle. \tag{20}$$

(ii) Their sum is determined by the relative update, $\alpha + \beta = \log u$ with $u = \|\Delta h\|_2/\|h\|_2$.

(iii) For any $\lambda > 0$, the pair $(g/\lambda, \lambda z)$ produces the same update as (g, z).

Proof. From Eq. 19, $\partial\Delta h/\partial\alpha = \partial\Delta h/\partial\beta = \Delta h$ at fixed h and $\hat{z}$. The chain rule gives (i). Taking norms in Eq. 19 gives $\|\Delta h\|_2 = \|h\|_2 e^{\alpha+\beta}$, which is (ii). For (iii), $(g/\lambda)(\lambda z) = gz$. □

Part (i) is Eq. 3 of the main text. The loss constrains only the sum $\alpha + \beta$, while any change of the split along $\alpha - \beta$ leaves the update and the loss unchanged. Part (iii) shows that a small gate is indistinguishable from a large gate combined with a small projection, so the gate value neither reflects nor bounds the injection strength. Whether the rescaling in (iii) can be realized by the parameters depends on the gate, which we examine next.

D.2 Saturation and Collapse of the Sigmoid Gate

Let $g = \sigma(a)$. Since $\partial \log g / \partial a = 1 - g$, Proposition 1(i) gives the gradient that reaches the gate logit,

$$\frac{\partial \mathcal{L}}{\partial a} = (1-g)\,\langle \nabla \mathcal{L}, \Delta h \rangle = (1-g)\,\frac{\partial \mathcal{L}}{\partial \beta}. \tag{21}$$

The gate and the projection thus receive the same signal on the magnitude of the update, attenuated by $1-g$ for the gate. Equivalently, $\partial \mathcal{L}/\partial a = g(1-g)\,\partial \mathcal{L}/\partial g$ with $\partial \mathcal{L}/\partial g = \langle \nabla \mathcal{L}, z \rangle$, which is the factor $g(1-g)$ in Section 3. The two forms differ only in the reference gradient, and both vanish as $g \to 1$. The two ends of the sigmoid lead to the two failure modes of Section 3.

Saturation. As $g \to 1$, the factor $1-g$ in Eq. 21 vanishes, and the magnitude signal flows almost entirely into the projection. For the Dot gate, which stays above 0.99 on 99.9% of tokens, the gate logit receives at most 1% of the gradient that reaches β. The gate then freezes into a constant $g \approx 1$, which W_O absorbs, so that $\Delta h \approx z$.

The Dot gate is prone to this regime because of the scale of its logit. For $x \in \mathbb{R}^D$, RMSNorm with learnable scale γ satisfies $\|\,\mathrm{RMSNorm}(x)\|_2 \le \|\gamma\|_\infty \sqrt{D}$. By the Cauchy–Schwarz inequality, the logits of Dot and Scaled Dot are bounded by

$$|a_{\text{Dot}}| \le \|\gamma_q\|_\infty \|\gamma_k\|_\infty \sqrt{D}, \qquad |a_{\text{Scaled Dot}}| \le \|\gamma_q\|_\infty \|\gamma_k\|_\infty. \tag{22}$$

With unit scales and $D = 2048$, the Dot logit can reach about 45 once q_g and k_g align, deep in the saturated regime, which is why its logit grows with $\sqrt{D}$. Dividing by D instead keeps the logit of Scaled Dot within a range set by the normalization scales. This prevents saturation, but also confines the gate to a narrow interval, consistent with its concentration near 0.2 in Figure 1.

Collapse. For the MLP gate, the logit is $a = W_2\ \mathrm{GELU}(W_1[h; r] + b_1) + b_2$, so the bias b_2 shifts the logit uniformly across all inputs. For $\lambda \ge 1$, consider the reparameterization

$$b_2 \mapsto b_2 - \log \lambda, \qquad W_O \mapsto \lambda W_O. \tag{23}$$

The new gate is $g' = \sigma(a - \log\lambda) = e^a/(\lambda + e^a)$, and the new update is $\Delta h' = g' \lambda z$. Its ratio to the original update is

$$\frac{\|\Delta h'\|_2}{\|\Delta h\|_2} = \frac{\lambda g'}{g} = \frac{\lambda(1+e^a)}{\lambda + e^a} \in \big[1,\ 1 + e^a\big), \tag{24}$$

with the same direction. Since $e^a = g/(1-g)$, the reparameterization changes the update by a relative amount below $g/(1-g)$ on every input, uniformly in λ. For the MLP gate with mean 0.0067, this is below 0.7%, and for MLP + Norm, with $g < 5 \times 10^{-4}$, it is below 0.05%. In this regime, Proposition 1(iii) is realized to high accuracy by a single parameter, b_2, together with the scale of W_O. The loss is nearly flat along this path, and the gate can slide along it toward an ever smaller gate and an ever larger projection without affecting the update.

D.3 Properties of Decoupled Fusion

We now prove the relations in Eq. 8. Recall that decoupled fusion computes

$$p = z - \frac{h^\top z}{\|h\|_2^2}\, h, \qquad d = \frac{p}{\|p\|_2}, \qquad \widetilde{h} = h + \rho g \|h\|_2\, d, \tag{25}$$

for $h \neq 0$ and $p \neq 0$. The degenerate case $p = 0$ is handled as in Appendix B.4.

Proposition 2 (Bounded update and rotation)**.** *Decoupled fusion satisfies*

$$u = \frac{\|\widetilde{h} - h\|_2}{\|h\|_2} = \rho g \le \rho, \qquad \angle(h, \widetilde{h}) = \arctan(\rho g), \qquad \|\widetilde{h}\|_2 = \|h\|_2 \sqrt{1 + \rho^2 g^2}. \tag{26}$$

Proof. First, $h^\top p = h^\top z - (h^\top z/\|h\|_2^2)\,\|h\|_2^2 = 0$, so $h^\top d = 0$, and $\|d\|_2 = 1$ by construction. Then $\|\widetilde{h} - h\|_2 = \rho g \|h\|_2 \|d\|_2 = \rho g \|h\|_2$, which gives the first relation, and $u \le \rho$ follows from $g < 1$. Since $d \perp h$, the Pythagorean theorem gives $\|\widetilde{h}\|_2^2 = \|h\|_2^2 + \rho^2 g^2 \|h\|_2^2$. Finally,

$$\cos \angle(h, \widetilde{h}) = \frac{h^\top \widetilde{h}}{\|h\|_2 \|\widetilde{h}\|_2} = \frac{\|h\|_2^2}{\|h\|_2^2 \sqrt{1 + \rho^2 g^2}} = \frac{1}{\sqrt{1 + \rho^2 g^2}}, \tag{27}$$

so $\tan \angle(h, \widetilde{h}) = \rho g$. □

The gate value can therefore be read directly from the update as $g = u/\rho$, and each value corresponds to a unique rotation angle. With $\rho = 0.7$, the rotation is at most $\arctan(0.7) \approx 35^\circ$ and the norm grows by at most a factor of $\sqrt{1+\rho^2} \approx 1.22$.

D.4 Scale Invariance and Gradient Decomposition

Proposition 2 shows that the update is bounded, but not yet that the projection can no longer take over its magnitude. We show this at the level of both parameters and gradients.

Proposition 3 (Scale invariance)**.** *For any $\lambda > 0$, replacing W_O by λW_O leaves d, g and Δh unchanged. Consequently, $\langle \nabla_{W_O} \mathcal{L}, W_O \rangle = 0$.*

Proof. Replacing W_O by λW_O maps z to λz and, by linearity of the projection, p to λp, so $d = p/\|p\|_2$ is unchanged. The gate reads (h, r) and does not depend on W_O, so g and hence Δh are unchanged. The loss is therefore invariant along $\lambda \mapsto \lambda W_O$, and differentiating $\mathcal{L}(\lambda W_O)$ at $\lambda = 1$ gives $\langle \nabla_{W_O} \mathcal{L}, W_O \rangle = 0$. □

Unlike Proposition 1(iii), no rescaling of W_O can be traded against the gate. The flat direction that allowed the gate to collapse no longer exists, since moving along it changes neither the gate nor the update.

Gradient decomposition. The same conclusion follows from how the gradient is distributed between the two branches. Treating $\|h\|_2$ and $h/\|h\|_2$ as constants, as in our implementation, and decomposing $\nabla \mathcal{L}$ into a component along d and a component orthogonal to it,

$$\frac{\partial \mathcal{L}}{\partial g} = \rho \|h\|_2 \, \langle \nabla \mathcal{L}, d \rangle, \qquad \frac{\partial \mathcal{L}}{\partial p} = \frac{\rho g \|h\|_2}{\|p\|_2} \left(I - dd^\top\right) \nabla \mathcal{L}, \tag{28}$$

where we used $\partial d / \partial p = (I - dd^\top)/\|p\|_2$. The gate receives exactly the component of the gradient along the update direction, which carries the information on how large the update should be. The projection receives only the orthogonal component, and $(I - dd^\top)$ removes any signal to lengthen or shorten the update. Compared with Eq. 21, the two branches no longer share the same magnitude signal. The magnitude is learned by the gate alone, and the projection learns only the direction.

Growth of the projection scale. Proposition 3 also explains why the projection scale of BIND-VLA grows freely in Figure 3(b) without affecting the update. Under a gradient step $W_O \mapsto W_O - \eta G$ with $G = \nabla_{W_O} \mathcal{L}$, orthogonality gives

$$\|W_O - \eta G\|_F^2 = \|W_O\|_F^2 + \eta^2 \|G\|_F^2 \geq \|W_O\|_F^2, \tag{29}$$

so the norm of W_O can only grow in the absence of weight decay, as is known for scale-invariant weights under normalization (Van Laarhoven, 2017). Our weight decay of 10^{-10} is negligible, and although AdamW does not follow the gradient direction exactly, we observe the same growth in practice. Since the update does not depend on this scale, the growth is harmless, and it is the reason why a large projection scale no longer indicates a large injection.

Remark. Decoupling removes the flat direction between the gate and the projection, but it does not prevent the loss itself from driving the gate toward an extreme. From Eq. 28, the gradient reaching the logit is $g(1-g)\,\rho \|h\|_2 \langle \nabla \mathcal{L}, d \rangle$, which becomes small as $g \to 0$. This is the intended behavior when memory does not help. On LIBERO, the gate of BIND-VLA decreases to between 10^{-14} and 10^{-12}, and, unlike the collapsed MLP gates of Section 3, the update vanishes with it, because no projection can compensate. The initialization $b_2 = -3$ of Appendix B.3 keeps the gate away from this regime at the start of training, so that it is reached only when driven by the loss.

Role of Orthogonalization Normalization alone, as in MLP + Unit, already gives $u = \rho g$. We now show why the resulting gate still does not correspond to a fixed change of the representation. Let $e = h/\|h\|_2$ and let θ be the angle between z and h. The unit direction decomposes as $\hat{z} = \cos\theta\, e + \sin\theta\, d$, with d the orthogonalized direction, and the Unit fusion gives

$$\widetilde{h} = \big(1 + \rho g \cos\theta\big)\, h + \rho g \sin\theta\, \|h\|_2\, d. \tag{30}$$

The first term only rescales h. The next backbone layer normalizes its input, and RMSNorm is invariant to positive rescaling up to its ϵ (Xiong et al., 2020), so this component is largely removed. What reaches the next layer is the direction of $\widetilde{h}$, which is rotated from h by

$$\tan\angle(h, \widetilde{h}) = \frac{\rho g \sin\theta}{1 + \rho g \cos\theta}. \tag{31}$$

The denominator is positive whenever $\rho g < 1$, which holds for all budgets used in this work, since $\rho \leq 0.99$ and $g < 1$. The angle is then well defined. The effective change caused by a given gate value, however, now depends on the alignment θ between the projection and the current representation, which varies across tokens and during training. When z is parallel to h, $\sin\theta = 0$ and the update leaves the direction of h untouched, regardless of g. At the other extreme, differentiating Eq. 31 with respect to θ shows that the rotation is largest at $\cos\theta = -\rho g$, where $\widetilde{h}$ is tangent to the sphere of radius $\rho g\|h\|_2$ around h and $\sin\angle(h, \widetilde{h}) = \rho g$. This maximum, $\arcsin(\rho g)$, exceeds the rotation $\arctan(\rho g)$ of decoupled fusion, because part of the update then also shortens h. The same gate value can thus correspond to any rotation between 0 and $\arcsin(\rho g)$, and the gate is identifiable from the update norm but not from its effect.

Orthogonalization fixes $\theta = \pi/2$ for every token, and Eq. 31 reduces to $\arctan(\rho g)$ from Proposition 2. This is not the largest rotation attainable for a given update norm, but it is the same for every token, so the gate determines both the norm of the update and the rotation it causes. No part of the update is spent on rescaling h. This is consistent with Table 2, where MLP + Unit falls 6.6 points below BIND-VLA despite sharing the same gate, the same budget and the same identity $u = \rho g$.

E Diagnostic Protocol

This section describes how the gate statistics in Figures 1, 3 and 4 are measured. All three figures use the same probing protocol, applied to different checkpoints.

E.1 Probing Protocol

A probe measures the memory injection of a checkpoint on observations from its training distribution. We draw 32 observations from the training set of the checkpoint with a fixed random seed of 42, skipping observations with fewer than three past timesteps in their history, whose memory bank is nearly empty. Each observation is passed through the policy once with batch size 1. At the injection layer, we record for every token of the prefix the gate g, the projection scale $s = \|z\|_2/\|h\|_2$ and the relative update $u = \|\widetilde{h} - h\|_2/\|h\|_2$, where s is measured before any normalization of z. Since memory is injected at a single layer, each token yields one value of each quantity.

We record all prefix tokens, including the padding positions of the language instruction. On MIKASA-Robo, the prefix consists of 256 top-view tokens, 256 wrist-view tokens and 200 language positions, so each probe yields $32 \times 712 = 22{,}784$ values per quantity. On LIBERO, the $\pi_{0.5}$ input additionally contains a third camera slot, which is filled with a blank image and contributes another 256 tokens, giving $32 \times 968 = 30{,}976$ values. The values of all 32 observations are pooled into a single set, from which we compute means, standard deviations and percentiles. Standard deviations are population standard deviations.

E.2 Gate Designs (Figure 1)

Figure 1 probes the five gate designs of Section 3, whose formulations are given in Appendix C. All five checkpoints are trained on the Remember tasks with the protocol of Appendix B.1 and probed at step 10,000. Boxes span the 25th to 75th percentiles, whiskers span the 10th to 90th percentiles, horizontal lines mark medians and diamonds mark means. Values beyond the whiskers are drawn

as outliers. Since the gate and the projection scale are shown on logarithmic axes and the relative update on a fixed range, some outliers fall outside the plotted range. These are not drawn at their values, but counted and marked by a triangle at the corresponding edge of the axis, with ▲ for values above the upper edge and ▼ for values below the lower edge. This is a plotting convention only and involves no outlier detection.

E.3 Training Dynamics (Figure 3)

Figure 3 probes Dot and BIND-VLA at intermediate checkpoints of training on the Remember tasks. Each point is one probe as described above. Panel (a) shows the coefficient of variation of the gate, and panels (b) and (c) show the means of s and u, with bands spanning the 10th to 90th percentiles over tokens. The bands describe the spread across tokens and are not confidence intervals.

Table 9 lists the probed steps. The main training runs saved checkpoints only at a few late steps. We therefore obtain the earlier checkpoints from a second run with the same configuration, and use the main runs for the remaining steps, including step 10,000, which corresponds to the models evaluated in the main text. Lines connect only the probed steps.

Table 9: **Probed checkpoints in Figure 3.**

Model	Second run with the same configuration	Main run
Dot	10, 50, 100–400 (every 50), 500–800 (every 100), 1,000, 3,000, 5,000, 7,000	10,000
BIND-VLA	10, 50, 100–400 (every 50), 500–800 (every 100), 1,000	5,000, 7,500, 10,000

E.4 Memory Injection across Benchmarks (Figure 4)

Figure 4 compares the injection of BIND-VLA on MIKASA-Robo and LIBERO, and of Dot on LIBERO. On MIKASA-Robo, we probe the BIND-VLA checkpoint evaluated on the Remember tasks. On LIBERO, we probe the checkpoint trained on each suite, drawing the 32 observations from the training set of that suite. All checkpoints are probed at step 10,000. The bars show the means of s and u over all tokens of the probe.

E.5 Coefficient of Variation

We measure the variability of the gate by its coefficient of variation over the pooled values of a probe,

$$\mathrm{CV}(g) = \frac{\mathrm{std}(g)}{|\,\mathrm{mean}(g)|}, \qquad \mathrm{std}(g) = \Big(\frac{1}{N}\sum_{i=1}^{N}(g_i - \bar{g})^2\Big)^{1/2}, \tag{32}$$

with $N = 22{,}784$ on MIKASA-Robo, and report it in percent. The CV is computed once over all tokens of all 32 observations, rather than per observation and then averaged, so it captures variation both within and across observations. Normalizing by the mean makes the CV comparable across gates of different magnitude. For a collapsed gate whose mean approaches zero, however, the small denominator dominates the CV, which then no longer reflects input-dependent modulation. We therefore report the CV only for Dot and BIND-VLA, whose gates do not collapse.

F Benchmark Details

F.1 MIKASA-Robo

MIKASA-Robo (Cherepanov et al., 2026) is a suite of tabletop manipulation tasks built on ManiSkill3 with a Franka Panda arm, designed so that the information needed for the final decision is available only in earlier observations. We use the VLA versions of its tasks (`*-VLA-v0`), which come with demonstrations for imitation learning and shorter time limits than the reinforcement learning versions of the original benchmark. Table 10 summarizes all tasks used in this work.

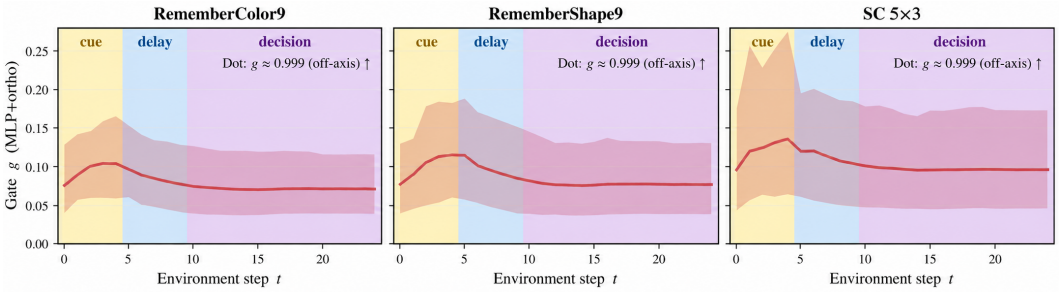


Figure 5: **Gate over the course of an episode.** Mean gate of BIND-VLA per environment step on three Remember tasks, averaged over 20 episodes, where SC5×3 denotes RememberShapeAndColor5×3. Bands show the 10th to 90th percentiles over tokens. The Dot gate stays at 0.999 throughout and lies above the plotted range. Shaded regions mark steps 0–4, 5–9 and 10 onward. The target is shown only within the first window, and every episode is in the decision phase from step 10 onward, but the exact phase boundaries vary across episodes.

Remember tasks. The nine Remember tasks of Table 2 follow a delayed match-to-sample structure. At the beginning of the episode, a single target object is shown on the table for a random number of steps, at most five. It then disappears, and the table stays empty for a random number of steps, again at most five. Immediately afterwards, K candidate objects appear, exactly one of which matches the target, and the episode succeeds when the policy touches the matching object. The candidates therefore appear no later than step 10, and the target is no longer visible when the decision is made. The three task families differ in which attribute must be remembered.

Additional tasks. To compare with prior work in Table 1, we also report two tasks with different memory requirements. In **ShellGameTouch**, a ball is shown at one of three positions at the beginning of the episode, after which all three positions are covered by cups. The policy must touch the cup that hides the ball. In **InterceptMedium**, a ball rolls across the table at a medium speed, and the policy must move into its path and deflect it toward a target. This requires estimating the motion of the ball from past observations. The benchmark also contains slow and fast variants that share the same instruction, so the speed must be inferred from the observations as well.

Chance levels. For the selection tasks, we define the chance level as $1/K$, the success rate of touching one of the K visible candidates uniformly at random. This is the standard chance level of delayed match-to-sample tasks. It assumes that the policy reaches a candidate, and a policy with random actions would perform far below it. InterceptMedium is a continuous control task without discrete candidates, and has no chance level in this sense.

Observations and actions. The policy observes RGB images from a fixed top camera and a wrist camera, rendered at 128×128 and resized to 224×224 before the visual encoder, together with the language instruction. The proprioceptive state is 7-dimensional, consisting of the end-effector position, its orientation as roll, pitch and yaw, and the gripper opening. Actions are 7-dimensional end-effector pose deltas with a gripper command, normalized to $[-1, 1]$, and the control frequency is 10 Hz. States and actions are zero-padded to the 32 dimensions of $\pi_{0.5}$.

Data. Each task comes with 250 successful demonstrations collected by a PPO agent with access to privileged state, released with the benchmark. We do not collect additional data. Models evaluated on the Remember tasks are trained jointly on the combined 2,250 demonstrations of all nine tasks, which contain 33,946 frames. Successful Remember demonstrations last 15.1 ± 2.1 steps on average, so the memory span covers the cue, the delay and a short reaching motion. For ShellGameTouch and InterceptMedium, we train a separate model on the 250 demonstrations of each task.

Evaluation. We evaluate every task on 100 episodes, instead of the 50 of the official evaluation client, with a fixed evaluation seed shared by all methods. Each episode runs until success or the time limit of the environment, which is 25 steps for the Remember tasks, 30 for ShellGameTouch

and 60 for InterceptMedium. The memory bank is cleared at the start of each episode, and averages over tasks are unweighted.

Table 10: **MIKASA-Robo tasks used in this work.** K is the number of candidates at decision time, $T_{\max}$ the time limit in steps, and length the mean and standard deviation of demonstration lengths in steps. All tasks use 250 demonstrations and the top and wrist cameras.

Task	Memorized attribute	K	Chance (%)	$T_{\max}$	Length
RememberColor3	color	3	33.3	25	15.4 ± 2.0
RememberColor5	color	5	20.0	25	15.3 ± 1.9
RememberColor9	color	9	11.1	25	15.1 ± 2.1
RememberShape3	shape	3	33.3	25	14.9 ± 2.2
RememberShape5	shape	5	20.0	25	15.1 ± 1.9
RememberShape9	shape	9	11.1	25	15.5 ± 2.1
RememberShapeAndColor3×2	shape and color	6	16.7	25	15.0 ± 2.1
RememberShapeAndColor3×3	shape and color	9	11.1	25	14.3 ± 2.0
RememberShapeAndColor5×3	shape and color	15	6.7	25	15.4 ± 2.2
ShellGameTouch	ball location	3	33.3	30	21.5 ± 5.2
InterceptMedium	ball motion	–	–	60	41.8 ± 8.4

Table 11: **Language instructions of the MIKASA-Robo tasks.** Tasks within each family share the same instruction.

Tasks	Instruction
RememberColor$K$	Observe the cube's color, wait, then touch the cube of the same color.
RememberShape$K$	Observe the object's shape, wait, then touch the object of the same shape.
RememberShapeAndColor$m \times n$	Observe the object's shape and color, wait, then touch the object of the same shape and color.
ShellGameTouch	Observe which cup hides the ball, wait, then touch that cup.
InterceptMedium	Intercept the rolling ball by moving to its path and deflecting it toward the target.

F.2 LIBERO

LIBERO (Liu et al., 2023) is a benchmark for language-conditioned manipulation with a Franka Panda arm in robosuite. We use its four standard suites, each consisting of 10 tasks.

- **LIBERO-Spatial**: the same objects in different spatial layouts. The policy must pick up the bowl specified by its spatial relation to other objects and place it on a plate.
- **LIBERO-Object**: the same layout with different objects. The policy must place the specified grocery item in a basket.
- **LIBERO-Goal**: the same objects and layout with different goals, such as opening a drawer or placing a bowl on the stove.
- **LIBERO-10**: ten long-horizon tasks across different scenes, each composed of multiple subtasks, such as placing two items in a basket.

Unlike MIKASA-Robo, these suites are not designed around memory, and the information needed to act is almost always available in the current observation and the instruction. We therefore use them to test whether attaching memory harms a policy that rarely needs history. The history, however, is long, with demonstrations lasting 122 to 268 steps on average depending on the suite.

Data. We train on the LIBERO demonstrations released with openpi, which contain 1,693 successful demonstrations over the 40 tasks. We train and evaluate a separate model for each suite. Table 12 lists the number of demonstrations and their lengths.

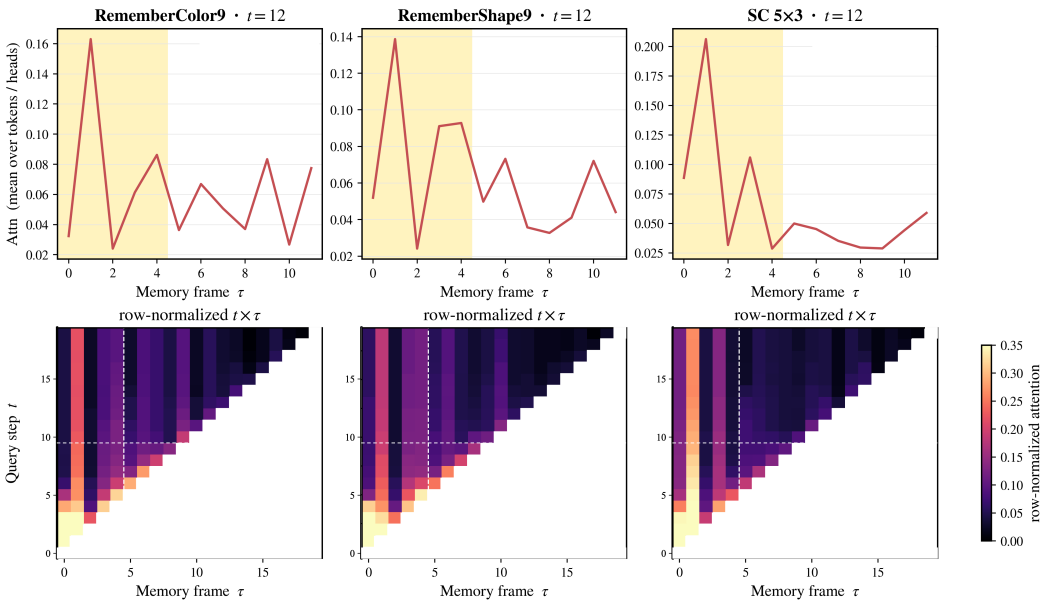


Figure 6: **Retrieval attention over the memory bank.** Top: attention of BIND-VLA over past timesteps τ at query step $t = 12$, averaged over heads, tokens and episodes. Bottom: attention over past timesteps $\tau < t$ at every query step t. The shaded region and the vertical dashed line mark past timesteps $\tau \leq 4$, which contain all cue frames, and the horizontal dashed line marks $t = 10$, from which every episode is in the decision phase. SC5×3 denotes RememberShapeAndColor5×3.

Observations and evaluation. The policy observes the third-person and wrist camera images, rendered at 256×256, rotated by $180°$ as in the training data and resized with padding to 224×224, together with the language instruction. The proprioceptive state consists of the end-effector position, its orientation in axis-angle form and the gripper state. Following the evaluation protocol of openpi, each episode starts with 10 idle steps to let objects settle, and uses the initial states provided by the benchmark. We evaluate 50 episodes per task, 500 per suite, with a fixed evaluation seed. For BIND-VLA, the memory bank is cleared at the start of each episode and accumulates one memory token per step, while the memory-free $\pi_{0.5}$ receives only the current observation.

Table 12: **LIBERO suites used in this work.** Length is the mean and standard deviation of demonstration lengths in steps, and $T_{\max}$ the time limit during evaluation.

Suite	Tasks	Demonstrations	Length	$T_{\max}$
LIBERO-Spatial	10	432	122.6 ± 20.8	220
LIBERO-Object	10	454	147.5 ± 18.5	280
LIBERO-Goal	10	428	121.6 ± 37.6	300
LIBERO-10	10	379	267.7 ± 56.7	520

G Visualization of Memory Gating

Section 5.3 characterizes the gate of BIND-VLA through statistics aggregated over tokens. Here we examine how the gate and the retriever behave within individual episodes. We roll out BIND-VLA and Dot on RememberColor9, RememberShape9 and RememberShapeAndColor5×3, the task with the most candidates in each Remember family, for 20 episodes each. At every environment step, we record the gate of every hidden-state token and the retrieval attention over the memory bank. To align these quantities with the environment steps, the policy replans at every step in these rollouts, instead of every 5 steps as in the main evaluation (Appendix B.6). As described in Appendix F.1, the target is shown for at most five steps, the table then stays empty for at most five steps, and the candidates appear immediately afterwards. We refer to these as the cue, delay and decision

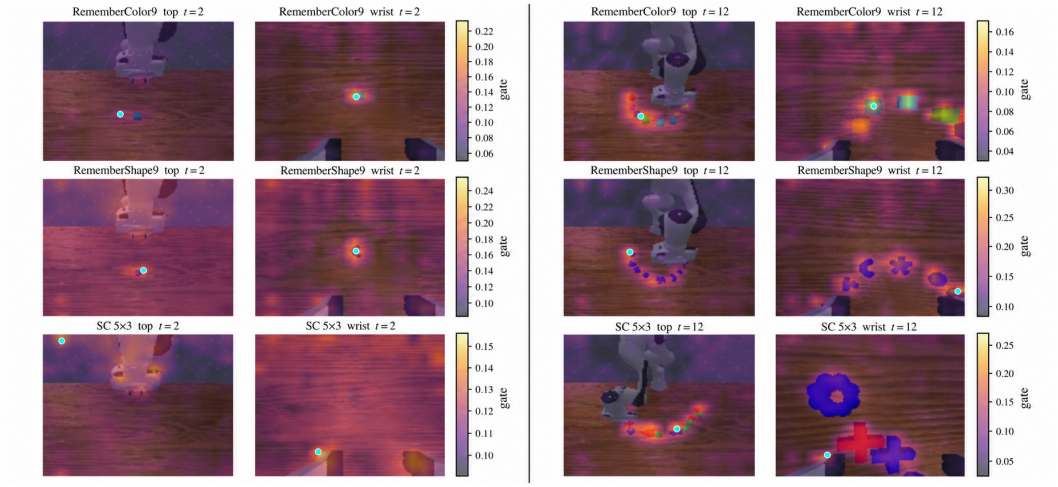


Figure 7: **Spatial distribution of the gate.** Gate of BIND-VLA on the top-view and wrist-view tokens at an early step ($t = 2$, left) and a decision step ($t = 12$, right) of successful episodes, upsampled from the 16×16 patch grid and overlaid on the input images. Cyan dots mark the patch with the highest gate. At each step, the two views of a task share one color scale. SC5×3 denotes RememberShapeAndColor5×3.

phases. Their boundaries vary across episodes, but every episode is in the decision phase from step 10 onward.

Gate over time. Figure 5 shows the gate at each step, averaged over tokens and episodes. The Dot gate stays at 0.999 at every step on all three tasks, with no visible variation across steps or tokens, consistent with the saturation identified in Section 3. The gate of BIND-VLA is an order of magnitude smaller and varies within the episode, following the same pattern on all three tasks. It rises over the first steps, when the target is shown, decreases over the following steps, and settles at a stable level from step 10 onward, when every episode is in the decision phase. Since the phase boundaries vary across episodes, the transitions in the averaged curves are gradual. Averaged over steps 0–4, 5–9 and 10 onward, the gate is 0.094, 0.085 and 0.071 on RememberColor9, 0.099, 0.096 and 0.077 on RememberShape9, and 0.120, 0.112 and 0.096 on RememberShapeAndColor5×3. At every step, the gate also spreads widely across tokens, as shown by the 10th to 90th percentile band.

The gate also differs across tasks. It is higher in every window on RememberShapeAndColor5×3, where both shape and color must be remembered, than on the other two tasks. At a much larger scale, the same gate is switched off on LIBERO, where its mean falls to between 10^{-14} and 10^{-12} (Section 5.3 and Figure 4), whereas it stays near 0.1 throughout the Remember episodes shown here. The gate of BIND-VLA thus depends on the task, the stage of the episode and the individual token, whereas the Dot gate depends on none of them.

Retrieval over history. Figure 6 shows the retrieval attention over past timesteps, averaged over heads, tokens and episodes. The retriever does not spread its attention uniformly over the history. From the first steps onward, a single early frame, $\tau = 1$, receives consistently high attention, which forms a persistent stripe across all later query steps and remains through the decision phase. During steps 5 to 9, the most recent frames also receive high attention, which fades from step 10 onward. At step 12, the frame $\tau = 1$ receives between 0.14 and 0.21 of the attention, 1.7 to 2.5 times the uniform level of $1/12$. Retrieval thus selects specific timesteps from the history, and the gate then determines how strongly the retrieved memory is injected.

Gate over tokens. Figure 7 maps the gate of the visual tokens back onto the 16×16 patch grid of each camera, at an early step ($t = 2$) and a late step ($t = 12$) of successful episodes. At the late step, the candidates are present, and high gate values in the top view follow the arc of candidate objects, while the background of the table remains low. The highest patch reaches three to four times the mean of its camera. At the early step, no candidates are present yet in the episodes shown.

In the episodes of RememberColor9 and RememberShape9, the target is still shown, and the gate concentrates on it, most clearly in the wrist view of RememberColor9, where the highest patch lies on the target cube at 3.3 times the camera mean. In the episode of RememberShapeAndColor5×3, no object is visible at this step, and the gate is nearly uniform across patches. We observe the same spatial pattern in failed episodes, so it is not specific to the selected examples. These examples suggest that the gate injects memory selectively into the tokens that describe task-relevant objects. This token-level dependence of the injection strength is part of what the Flattened intervention of Table 5 removes, which lowers the average success of BIND-VLA by 9.2 points.

H Additional Ablations

This section provides additional ablations that complement Table 4.

H.1 Injection Layer

Table 4(b) compares three injection layers under the default configuration. To examine the effect of depth at a finer granularity, Table 13 reports an earlier sweep over five layers, conducted during the architecture exploration under a different configuration. It uses encoder compression with pooled retrieval, the dot-product gate with residual fusion, and a second training stage of 30k steps on 8 GPUs. Absolute success rates are therefore not comparable with Table 4. Injecting after the first or second layer gives similar results, 31.8% and 31.6%, and success then decreases steadily with depth, to 29.1%, 24.8% and 22.3% at layers 5, 8 and 11. The decrease appears on most tasks and is consistent with the trend in Table 4(b), where early injection performs best.

Table 13: **Effect of the injection layer under pooled retrieval.** Success rates (%) on the nine Remember tasks, with encoder compression, pooled retrieval, the dot-product gate and residual fusion, trained for 30k steps on 8 GPUs. Layer ℓ denotes injection after the ℓ-th backbone layer. SC$m\times n$ denotes RememberShapeAndColor$m\times n$.

Layer	Color3	Color5	Color9	Shape3	Shape5	Shape9	SC3×2	SC3×3	SC5×3	Avg.
1	**58**	41	**27**	52	**19**	14	39	16	**20**	**31.8**
2	56	**46**	20	**54**	17	19	**42**	16	14	31.6
5	51	37	20	42	**19**	**22**	35	**18**	18	29.1
8	40	33	18	42	**19**	15	35	13	8	24.8
11	39	32	15	47	**19**	12	18	8	11	22.3

H.2 Update Budget

Table 4(c) reports results for three update budgets. Table 14 extends this analysis to six budgets, retraining BIND-VLA for each value with the same initialization scheme and all other settings fixed. Since the gate bias is initialized to $b_2 = -3$ for every budget, the initial relative update $\rho\,\sigma(-3)$ scales with ρ. The default budget, $\rho = 0.7$, achieves the highest average success rate of 38.4%, outperforming the next-best budget, $\rho = 0.5$, by 2.4 percentage points.

Average success increases with the update budget up to $\rho = 0.7$ and drops sharply at $\rho = 0.99$. A small budget caps the largest update that memory can apply, which limits tasks that need strong recall. A large budget makes the update more sensitive to the gate, since the same change in g moves the representation by ρ times as much, and the drop at $\rho = 0.99$ suggests that this sensitivity outweighs the larger capacity. This is consistent with the discussion of the update budget in Section 5.4.

I Qualitative Examples

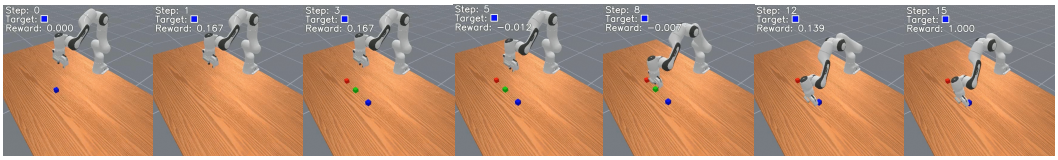
We show rollouts of BIND-VLA on the nine Remember tasks and on LIBERO-10, using the checkpoints and evaluation protocol of the main experiments (Appendix B.6). For each task, we show one successful episode. Each row shows one episode, with frames ordered in time from left to right.

Table 14: **Sensitivity to the update budget ρ.** Success rates (%) of BIND-VLA on the nine Remember tasks. SC$m\times n$ denotes RememberShapeAndColor$m\times n$. The row $\rho = 0.7$ corresponds to the default configuration.

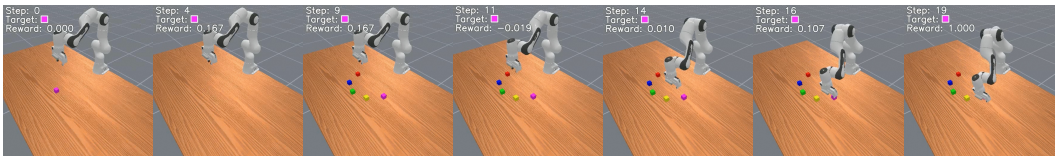
ρ	Color3	Color5	Color9	Shape3	Shape5	Shape9	SC3×2	SC3×3	SC5×3	Avg.
0.10	56	59	24	39	22	14	40	15	15	31.6
0.20	57	58	25	40	24	15	41	17	12	32.1
0.30	57	58	27	44	27	18	44	19	20	34.9
0.50	56	57	28	49	30	21	47	22	14	36.0
0.70	50	56	30	58	35	24	52	25	16	38.4
0.99	51	46	24	33	19	14	33	8	12	26.7

Remember tasks. Figures 8–10 show one episode of each Remember task. The tasks follow the delayed match-to-sample structure described in Appendix F.1. Since the durations of the cue and the delay vary across episodes, the frames of each row are chosen from the phases of that episode.

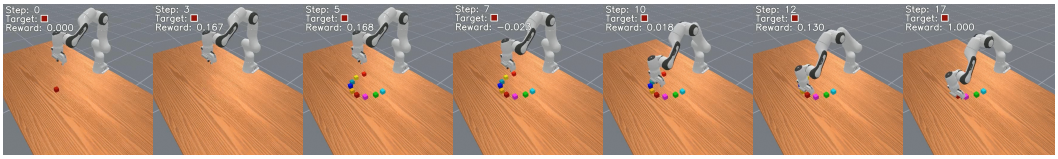
LIBERO. Figure 11 shows three episodes from LIBERO-10, whose tasks consist of multiple subtasks, such as placing two objects in a basket or placing a bowl in a drawer and closing it. Unlike the Remember tasks, these suites rarely require history, and the policy can act mostly on the current observation. As analyzed in Section 5.3, the gate of BIND-VLA switches memory injection off on these suites, and the policy performs on par with the memory-free $\pi_{0.5}$.



(a) RememberColor3 Manipulation

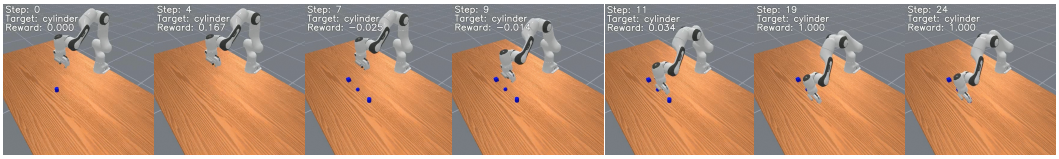


(b) RememberColor5 Manipulation

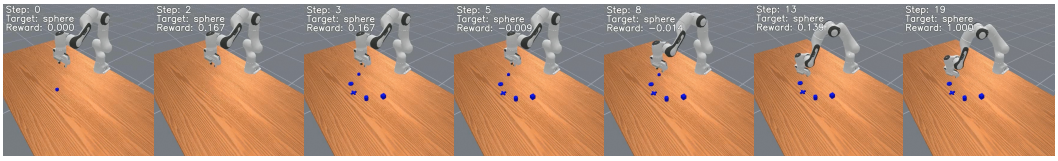


(c) RememberColor9 Manipulation

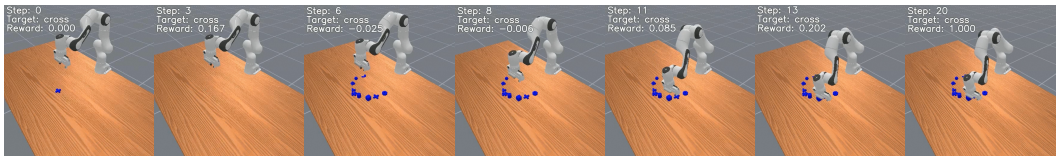
Figure 8: **Rollouts on the RememberColor tasks.** Rows show RememberColor3, RememberColor5 and RememberColor9 from top to bottom. Frames from the top camera cover the cue, delay and decision phases of each episode and its final step.



(a) RememberShape3 Manipulation

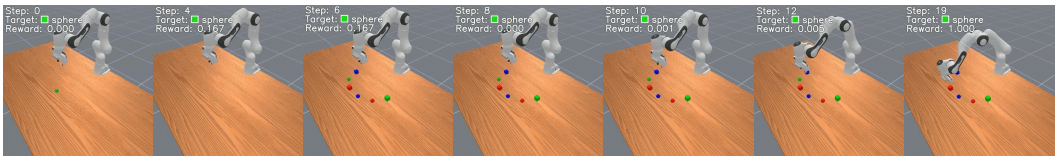


(b) RememberShape5 Manipulation

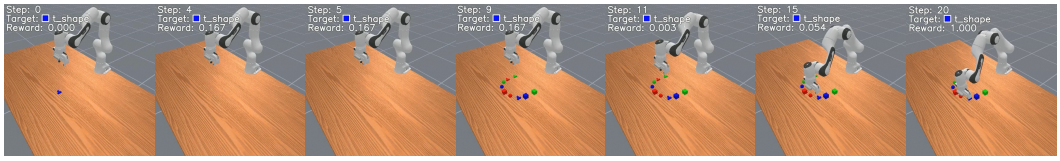


(c) RememberShape9 Manipulation

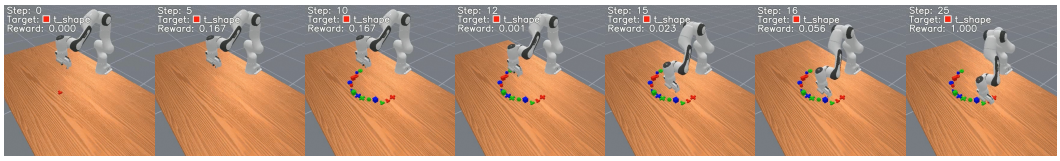
Figure 9: **Rollouts on the RememberShape tasks.** Rows show RememberShape3, RememberShape5 and RememberShape9 from top to bottom. Frames are rendered as in Figure 8.



(a) RememberShapeAndColor3x2 Manipulation



(b) RememberShapeAndColor3x3 Manipulation

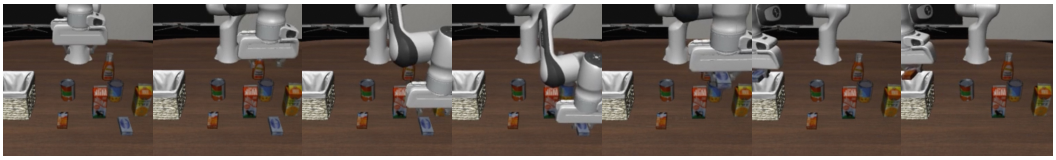


(c) RememberShapeAndColor5x3 Manipulation

Figure 10: **Rollouts on the RememberShapeAndColor tasks.** Rows show RememberShapeAndColor3×2, 3×3 and 5×3 from top to bottom. Frames are rendered as in Figure 8.



(a) Put both the alphabet soup and the tomato sauce in the basket



(b) Put both the cream cheese box and the butter in the basket



(c) Put the black bowl in the bottom drawer of the cabinet and close it

Figure 11: **Rollouts on LIBERO-10.** Each row shows one episode from the third-person camera, with the language instruction below it.