The *Vision Wormhole:* Latent-Space Communication in Heterogeneous Multi-Agent Systems

Xiaoze Liu[1,*], Ruowang Zhang[1,2,*], Weichen Yu[3], Siheng Xiong[4], Liu He[1]
Feijie Wu[1], Hoin Jung[1], Matt Fredrikson[3], Xiaoqian Wang[1], Jing Gao[1]
[1]Purdue University [2]Contextual AI [3]Carnegie Mellon University
[4]Georgia Institute of Technology [*]Equal contribution.
{xiaoze,zhan5763,he425,wu1977,jung414,joywang,jinggao}@purdue.edu
{weichenyu,mfredrik}@cmu.edu sxiong45@gatech.edu

Abstract

Multi-Agent Systems (MAS) powered by Large Language Models have unlocked advanced collaborative reasoning, yet they remain shackled by the inefficiency of discrete text communication, which imposes significant runtime overhead and information quantization loss. While latent state transfer offers a high-bandwidth alternative, existing approaches either assume homogeneous sender–receiver architectures or rely on pair-specific learned translators, limiting scalability and modularity across diverse model families with disjoint manifolds. In this work, we propose the **Vision Wormhole**, a novel framework that repurposes the visual interface of Vision-Language Models (VLMs) to enable model-agnostic, text-free communication. By introducing a Universal Visual Codec, we map heterogeneous reasoning traces into a shared continuous latent space and inject them directly into the receiver's visual pathway, effectively treating the vision encoder as a universal port for inter-agent telepathy. Our framework adopts a hub-and-spoke topology to reduce pairwise alignment complexity from $O(N^2)$ to $O(N)$ and leverages a label-free, teacher-student distillation objective to align the high-speed visual channel with the robust reasoning patterns of the text pathway. Extensive experiments across heterogeneous model families (e.g., Qwen-VL, Gemma) demonstrate that the Vision Wormhole reduces end-to-end wall-clock time in controlled comparisons while maintaining reasoning fidelity comparable to standard text-based MAS. Code is available at https://github.com/xz-liu/heterogeneous-latent-mas

1 Introduction

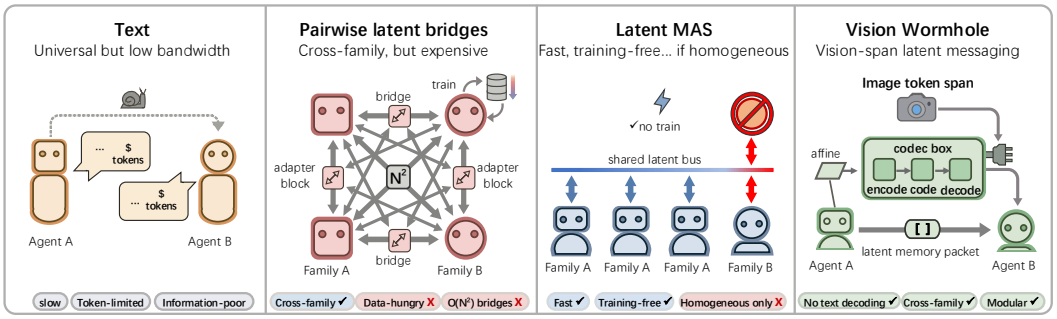

Figure 1: **The Vision Wormhole:** We repurpose the visual interface of Vision-Language Models (VLMs) to enable text-free communication across heterogeneous Multi-Agent Systems (MAS).

The field of Multi-Agent Systems (MAS) has evolved into complex societies of Large Language Models (LLMs) capable of collaborative reasoning (Guo et al., 2024; Tran et al., 2025; Yan et al., 2025), using distinct role assignments to decompose tasks and enhance performance (Wu et al., 2024; Hong et al., 2023; Li et al., 2023a; Zhang et al., 2024b). However, the reliance on discrete text communication (Yan et al., 2025) imposes a severe bottleneck, where decoding high-dimensional states into tokens incurs substantial runtime overhead and quantization error. While recent efforts in latent communication (Zou et al., 2025) attempt to bypass this by exchanging internal states like hidden activations or KV caches (Fu et al., 2025; Ye et al., 2025; Zheng et al., 2025), these approaches are largely restricted to homogeneous settings or require expensive pairwise translation modules. Such constraints fundamentally hinder the potential of heterogeneous MAS to combine diverse model strengths, such as specialized reasoning with generalist creativity.

Enabling latent communication across heterogeneous model families faces three fundamental challenges that existing approaches fail to address effectively:

The "Off-Manifold" Incompatibility. Unlike homogeneous models that share identical architectures, heterogeneous models (e.g., Qwen vs. Llama) operate on disjoint latent manifolds with incompatible dimensions and semantic geometries. A naive approach might employ a simple linear mapping to align these spaces. However, for standard text-only LLMs, this fails due to the "off-manifold" problem (Minixhofer et al., 2024; Feher et al., 2025; Minixhofer et al., 2025; Park et al., 2023b). Text embeddings are inherently discrete and sparse; a text-only LLM is trained solely on these discrete tokens and has never encountered arbitrary continuous vectors. Consequently, injecting a mapped, continuous vector directly into a text transformer typically destabilizes generation, as the input lies outside the model's valid data distribution, often leading to generation collapse.

The $O(N^2)$ Scalability Trap. To overcome manifold mismatches, recent work such as Cache-to-Cache (Fu et al., 2025) employs learned translation, training a neural fuser to project a sender's KV-cache into a receiver's space. While effective for specific pairs, this approach scales poorly in a diverse ecosystem of N agents. Establishing pairwise connections requires training $N(N-1)$ specific adapters, creating a quadratic complexity barrier. Furthermore, these translators are often non-trivial networks rather than lightweight modules.[1] This incurs substantial deployment costs and prevents the scalable creation of a general-purpose, plug-and-play latent MAS.

Absence of Aligned Supervision. Unlike text translation, where parallel corpora abound, there exists no natural ground-truth dataset pairing "Model A's hidden state" with "Model B's hidden state." Existing methods often rely on distilling from massive amounts of data (e.g., 500k samples for Cache-to-Cache) or end-to-end reinforcement learning, which is notoriously unstable. This lack of aligned supervision makes training a robust communication channel difficult without resorting to expensive, task-specific human annotation.

In this work, we identify a *natural* communication pathway that bypasses these limitations: the visual interface of Vision-Language Models (VLMs). Unlike text-only models, VLMs are explicitly trained to accept continuous, dense vectors via their visual encoders (Fein-Ashley & Fein-Ashley, 2025; Li et al., 2025a; Wang et al., 2025a). Recent work has also shown that the visual pathway can effectively *compress* and process textual information, e.g., by rendering chain-of-thought, documents, or code into visual form and leveraging the VLM's visual encoder for downstream reasoning and understanding (Wang et al., 2026; Wei et al., 2025; Shi et al., 2026). What we aim to do, however, is more: we hypothesize that the *vision-token input spaces* of different VLM families are naturally aligned (not only their text outputs), and can therefore serve as bridges between heterogeneous backbones.

The "image soft embedding" is, by definition, a fixed-length sequence of continuous variables that the model is conditioned to interpret as meaningful context. We hypothesize that this pre-existing pathway can be repurposed to transmit dense reasoning information between disjoint model families without the need to fine-tune the backbone parameters, yielding three key properties: (1) **Lightweight:** a small codec (e.g., $\sim$0.05B parameters)

[1] For instance, the adapters for translating Qwen3-0.6B $\rightarrow$ Qwen2.5-0.5B occupy 818.4 MB, comparable to the 988 MB backbone itself.

trained with a small amount of data, rather than a large, pair-specific translator; (2) **Modular:** once trained, codecs can be plugged in to connect arbitrary combinations of already-supported agents in an MAS without retraining pairwise adapters; and (3) **Bounded:** we can fix both the number of latent inference steps and the message bandwidth by mapping into a fixed-size visual token space, avoiding unbounded KV-cache translation where runtime can grow and errors may accumulate over long exchanges. We term this novel communication channel the *Vision Wormhole*. A wormhole is a speculative structure linking disparate points in spacetime, acting as a tunnel that allows instantaneous passage between distant locations or even separate universes (Einstein & Rosen, 1935). Analogously, our method bridges the universes of two completely heterogeneous neural networks. By encoding the reasoning traces of a sender model into a format that mimics the statistical properties of visual tokens, we create a tunnel through the VLM's vision encoder. This allows high-bandwidth, model-agnostic information transfer, effectively treating the visual pathway not just as an eye for the physical world, but as a universal port for model-to-model telepathy.

Contributions. To address these challenges, we propose **The Vision Wormhole**, a novel framework that repurposes the vision pathway of VLMs for text-free agent collaboration. Our contributions are four-fold:

- **The Vision Wormhole Mechanism:** We propose a paradigm shift by treating the VLM's vision encoder not as a sensory organ, but as a robust *communication interface*. By injecting latent information through the *image soft embedding* pathway, we bypass the discrete bottleneck of the text tokenizer. This exploits the VLM's native capability to process continuous signals, effectively resolving the Off-Manifold incompatibility that plagues text-only LLMs.
- **A Universal Codec for Heterogeneity ($O(N)$ Scalability):** We introduce a *Universal Latent Space* ($\mathcal{U}$) that acts as a standardized intermediate manifold. By adopting a "Hub-and-Spoke" topology, we map diverse model reasoning traces into this shared space before decoding them for the receiver. This design decouples the sender and receiver, reducing alignment complexity from quadratic $O(N^2)$ to linear $O(N)$, enabling new models to join the system by training only a single lightweight adapter.
- **Label-Free, Distillation-Based Alignment:** We develop a self-supervised distillation training objective that requires *no human annotation*. We treat the slow-but-accurate text-based communication as the "Teacher" and force the high-speed vision wormhole (the "Student") to mimic the teacher's internal representations and output distributions. This ensures the latent channel achieves high fidelity while remaining intuitive and plug-and-play.
- **Extensive Experimental Validation:** We conduct comprehensive experiments across heterogeneous model families. Results demonstrate that our method reduces overall MAS wall-clock time compared to text-based baselines while maintaining reasoning performance comparable to the original text-based MAS, validating the efficacy of the Vision Wormhole as a practical acceleration layer.

2 Preliminaries

Multi-agent system (MAS). We define a multi-agent system as $\mathcal{S} = (\mathcal{A}, \pi)$, where $\mathcal{A} = \{a_1, \ldots, a_N\}$ is a set of agents and π is the orchestration policy. Given an input query q, π specifies role execution, interaction order, and routing among agents, and the system outputs a final answer a. This definition is modality-agnostic: agents may communicate with discrete text messages (TextMAS) or via other interfaces when available. In our experiments, each agent a_i is a vision-language model (VLM) with frozen backbone F_i and input embedding dimension d_i. For an embedding sequence $X_i \in \mathbb{R}^{L \times d_i}$, agent i produces hidden states $H_i \in \mathbb{R}^{L \times d_i}$. We focus on heterogeneous MAS, where agent backbones may come from different model families. Relative to text-only communication, VLM agents expose an additional continuous interface via *visual token embeddings*, which we exploit as the communication channel (see Appendix D.1).

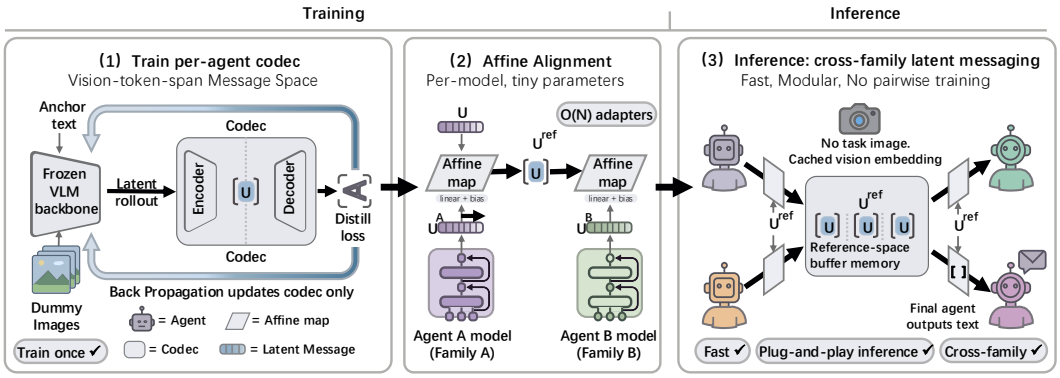

Figure 2: **Vision Wormhole overview.** Each agent 1) extracts a bounded latent rollout, encodes it into a fixed-size set of universal tokens, 2) aligns tokens through a shared reference space, and 3) decodes them into an injected perturbation written into the receiver's vision-token span.

The VLM visual interface as a continuous channel. A standard VLM forms an input embedding sequence by concatenating (1) text token embeddings and (2) a dedicated *image-token span*. Given an image $\mathbf{I}$, a vision encoder $E_{\text{vis}}^{(i)}$ and projector ϕ_i produce $X_{\text{img}}^{(i)} = \phi_i(E_{\text{vis}}^{(i)}(\mathbf{I})) \in \mathbb{R}^{L_{\text{img}}^{(i)} \times d_i}$, which is inserted into the language stream at model-specific image positions. Crucially, the VLM backbone F_i is trained to treat $X_{\text{img}}^{(i)}$ as valid semantic context, i.e., it already operates on a dense, continuous embedding manifold. This stands in contrast to text-only LLMs, whose training distribution contains only discrete token embeddings and is therefore brittle to arbitrary continuous inputs (the off-manifold problem).

Latent rollouts as a model-internal summary. Let a VLM backbone F_i process a prompt and produce the final hidden vector at the prompt boundary, $h_0^{(i)} \in \mathbb{R}^{d_i}$. We define a *latent rollout* by repeatedly feeding back a single continuous pseudo-token embedding derived from the previous hidden state while reusing the prompt's attention cache (detailed in Appendix C.1). At step t, we form an input embedding $x_t^{(i)} = \text{NormMatch}_i(h_t^{(i)})$, where NormMatch_i rescales vectors to match the typical norm of the model's token embeddings (see Eq. 9 in Appendix C.1). A T-length rollout yields $H_i = [x_0^{(i)}, \ldots, x_{T-1}^{(i)}] \in \mathbb{R}^{T \times d_i}$, which serves as the sender's continuous message substrate.

3 Method: The Vision Wormhole

We propose **The Vision Wormhole**: a latent communication layer that transmits information between heterogeneous agents by writing a continuous message into the *vision-token span* of a VLM. At a high level, each agent a_i is augmented with a lightweight *vision codec* (trained once, then frozen at inference) that: (i) extracts a short model-internal summary from the agent as a latent rollout, (ii) compresses it into a small, fixed set of *universal tokens*, (iii) maps these tokens into a shared *reference universal space* $\mathcal{U}$ via an affine alignment, and (iv) decodes received universal tokens into a perturbation that is injected into the agent's image-token span. All VLM backbone parameters remain frozen. Detailed codec, alignment, and inference protocols are provided in Appendix C. Figure 2 describes the overall pipeline.

Notation. For agent i, let d_i denote its embedding dimension. We use a universal token dimension D shared across all agents. A message is represented by $K_{\text{u}} = K + 2$ universal tokens (with K semantic tokens plus two special tokens: a global token and a style token).

For vision-span writing, we decode to K_{img} *image query tokens*. We denote by $L_{\text{img}}^{(i)}$ the number of image tokens used by agent i in its prompt.

3.1 Training a Vision Codec for a Specific VLM

We first train a per-agent codec that maps the agent's latent rollout to an injected vision-span embedding such that the frozen VLM behaves *as if* it had received the same content via text. This is done with *label-free self-distillation*: text-based communication acts as a teacher, and the vision wormhole acts as a student (distillation boundary details in Appendix C.1).

(1) Sender message extraction via latent rollout. Given a prompt (task context, role instructions, and any received messages), the backbone produces a boundary hidden state $h_0^{(i)}$. We then generate a length-T latent rollout $H_i = [x_0^{(i)}, \ldots, x_{T-1}^{(i)}] \in \mathbb{R}^{T \times d_i}$, where $x_t^{(i)} = \text{NormMatch}_i(h_t^{(i)})$ is a norm-calibrated pseudo-token embedding (Eq. 9 in Appendix C.1). Intuitively, H_i captures a short, model-internal continuation of the prompt in continuous space, acting as a compact summary of the agent's current reasoning state.

(2) Universal-token encoder. We compress H_i into a fixed-size set of universal tokens using a Perceiver-style resampler (cross-attention from a small set of learned queries to the rollout; details in Appendix C.1). Formally, the encoder $\mathcal{E}_i$ outputs $U_i = \mathcal{E}_i(H_i) \in \mathbb{R}^{K_u \times D}$. The K semantic tokens carry the message content, while two special tokens provide global aggregation and style/statistics cues (Eq. 13 in Appendix C.1). This design yields a *bounded* message representation whose size does not grow with text length.

(3) Universal-to-vision decoder and gated injection. The decoder $\mathcal{D}_i$ maps universal tokens to a vision-span perturbation and a scalar gate:

$$(\Delta_i, g_i) = \mathcal{D}_i(U_i), \qquad \Delta_i \in \mathbb{R}^{K_{\text{img}} \times d_i}, \;\; g_i \in (0, 1). \tag{1}$$

The perturbation Δ_i is a *continuous prompt* expressed in the agent's embedding space; the gate g_i allows the model to modulate injection strength per example. To keep the injected embeddings near the VLM's visual manifold, we inject *residually* relative to a fixed baseline visual embedding. Let $\bar{X}_{\text{img}}^{(i)} \in \mathbb{R}^{L_{\text{img}}^{(i)} \times d_i}$ denote the image-token embeddings induced by a fixed dummy image under the frozen VLM. We resample Δ_i to the required image-span length via a length-resampling operator $\text{Resample}(\cdot; L)$ and write

$$X_{\text{img}}^{(i)} \;=\; \bar{X}_{\text{img}}^{(i)} \;+\; g_i \cdot \text{Resample}(\Delta_i; L_{\text{img}}^{(i)}). \tag{2}$$

All downstream computation is performed by the frozen backbone conditioned on $X_{\text{img}}^{(i)}$. Decoder/gating and dummy-image resampling details are provided in Appendix C.1.

(4) Label-free distillation objective. Training uses anchor messages m (short text strings). The **teacher** prompt includes m explicitly as text. The **student** prompt omits m but contains a dummy image whose image-token span is overwritten by Eq. (2) computed from the teacher-side rollout. We optimize codec parameters (only $\mathcal{E}_i, \mathcal{D}_i$) to match teacher and student behavior under the frozen backbone.

Let h^{text} and ℓ^{text} denote the teacher hidden state and next-token logits at the prompt boundary, and let $h^{\text{vis}}, \ell^{\text{vis}}$ be the corresponding student quantities under vision injection. We minimize

$$\begin{aligned} \mathcal{L}_{\text{codec}} = \lambda_h \left\| h^{\text{vis}} - \text{stopgrad}(h^{\text{text}}) \right\|_2^2 \\ + \lambda_{\text{kl}} \, \tau^2 \, \text{KL}\Big(\text{softmax}\Big(\tfrac{\ell^{\text{text}}}{\tau}\Big) \;\Big\|\; \text{softmax}\Big(\tfrac{\ell^{\text{vis}}}{\tau}\Big) \Big) \\ + \lambda_{\text{rms}} \Big(\text{RMS}(\Delta_{\text{inj}}) - \text{RMS}(\bar{X}_{\text{img}}^{(i)}) \Big)^2, \end{aligned} \tag{3}$$

where τ is the distillation temperature and Δ_{inj} denotes the gated perturbation before resampling. The first two terms enforce representational fidelity and output-distribution fidelity, while the RMS matching stabilizes injection magnitude in the visual embedding manifold.

3.2 Aligning Codecs Across Agents via an Affine Map in Universal Space

Training the codec independently for each agent yields universal tokens U_i that live in a shared *dimension* D but not necessarily a shared *coordinate system*. To enable communication among N heterogeneous agents without training $O(N^2)$ pairwise translators, we adopt a hub-and-spoke design: each agent learns an affine map to and from a *reference* universal space $\mathcal{U}$. Additional motivation and closed-form fitting details are provided in Appendix C.2 and Appendix C.2.

Reference-space maps. Fix a reference agent r. For each agent i, we learn two affine maps:

$$U^{\text{ref}} = A_i^{\text{out}}(U_i) = U_i W_i^{\text{out}} + \mathbf{1}\,(b_i^{\text{out}})^\top, \tag{4}$$

$$U_i = A_i^{\text{in}}(U^{\text{ref}}) = U^{\text{ref}} W_i^{\text{in}} + \mathbf{1}\,(b_i^{\text{in}})^\top, \tag{5}$$

where $W_i^{\text{out}}, W_i^{\text{in}} \in \mathbb{R}^{D\times D}$ and biases $b_i^{\text{out}}, b_i^{\text{in}} \in \mathbb{R}^D$. This yields $O(N)$ alignment parameters (one map per model to the hub and one map from the hub), rather than $O(N^2)$ pairwise adapters.

Ridge regression from a small anchor set. We fit these affine maps using a small set of shared anchor texts $\{m_j\}_{j=1}^M$. For each anchor m_j, we compute universal tokens $U_i(m_j)$ for every model i using the already trained encoder. We then solve a regularized least-squares problem in closed form (ridge regression) to map each model's tokens to the reference tokens:

$$\min_{W_i^{\text{out}}, b_i^{\text{out}}} \sum_{j=1}^{M} \left\| U_i(m_j) W_i^{\text{out}} + \mathbf{1}(b_i^{\text{out}})^\top - U_r(m_j) \right\|_F^2 + \lambda \|W_i^{\text{out}}\|_F^2, \tag{6}$$

and analogously for A_i^{in}. Because the encoder already compresses messages into a small, structured token set, we empirically find that only a modest anchor set is required to align models. We discuss why such affine alignment is plausible in Appendix D.

3.3 Inference: Multi-Agent Collaboration through the Vision Wormhole

At inference time, agents collaborate by exchanging *only* universal tokens in the reference space. No intermediate text messages are generated; only the final agent emits the natural-language answer. The unified "read–think–write" abstraction, scheduling variants, and bounded communication-cost discussion are detailed in Appendix C.3, Appendix C.3, and Appendix C.3.

Message passing operator. A wormhole message from sender s to receiver i is computed as:

$$U_{s\to i}^{\text{ref}} = A_s^{\text{out}}\Big(\mathcal{E}_s(H_s)\Big), \qquad (\Delta_i, g_i) = \mathcal{D}_i\Big(A_i^{\text{in}}(U_{s\to i}^{\text{ref}})\Big), \tag{7}$$

followed by writing Eq. (2) into receiver i's image-token span.

Memory aggregation. Let $\mathcal{M}$ be the set of received messages for an agent, each stored as universal tokens in the reference space. Aggregation is necessary because a receiver typically gets multiple partial updates from different upstream roles (and possibly multiple rounds), each carrying complementary evidence. Combining them before decoding provides one coherent context for the next backbone call, instead of repeatedly decoding and re-running the model message-by-message. We aggregate memory by concatenation in token dimension:

$$U_{\text{mem}}^{\text{ref}} = \text{Concat}\big(\{U_m^{\text{ref}}\}_{m\in\mathcal{M}}\big) \in \mathbb{R}^{(|\mathcal{M}|\cdot K_{\text{u}})\times D}, \tag{8}$$

and decode a single vision-span perturbation from $U^{\text{ref}}_{\text{mem}}$. This implements a fixed-cost "read" from memory: regardless of how verbose a sender would have been in text, the receiver reads a bounded-size continuous context.

Role interaction loop. Consider a role-structured MAS with roles (e.g., planner, critic, refiner, solver/judger). Each non-final role runs the VLM once to produce a latent rollout message; the final role generates the answer text. Concretely, for each role-agent i in an ordered collaboration:

1. **Read:** Decode the current memory $U^{\text{ref}}_{\text{mem}}$ into a vision-span injection for agent i and run the frozen backbone conditioned on this injection.
2. **Think (latent):** Extract a rollout H_i and encode it into a new universal message $U^{\text{ref}}_i = A^{\text{out}}_i(\mathcal{E}_i(H_i))$.
3. **Write:** Append U^{ref}_i to the shared memory buffer.

Unlike directly injecting arbitrary continuous vectors into a text-only transformer, the Vision Wormhole writes into the VLM's image-token span, which is explicitly trained to accept dense continuous embeddings. Moreover, residual writing relative to $\bar{X}^{(i)}_{\text{img}}$ keeps the injected context near the visual embedding manifold, improving stability while still permitting high-bandwidth transfer.

4 Experiments

4.1 Experimental Settings

Tasks and Datasets. We follow the evaluation suite used in LatentMAS (Zou et al., 2025) and consider nine benchmarks spanning general and reasoning-intensive tasks: (i) *Math & Science Reasoning*, including GSM8K (Cobbe et al., 2021), AIME 2024 (Maxwell-Jia, 2024), AIME 2025 (math ai, 2025), GPQA (Rein et al., 2023), and MedQA (Yang et al., 2024a); (ii) *Commonsense Reasoning*, including ARC-Easy and ARC-Challenge (Clark et al., 2018b;a); and (iii) *Code Generation*, including MBPP-Plus and HumanEval-Plus (Liu et al., 2023b). Unless otherwise specified, we report accuracy for multiple-choice and short-answer tasks, and pass@1 for code-generation benchmarks.

Models. We evaluate heterogeneous MAS instantiated with off-the-shelf open-source backbones from multiple model families: Qwen/Qwen3-VL-2B-Thinking, Qwen/Qwen3-VL-4B-Thinking, Qwen/Qwen3-VL-8B-Thinking (Bai et al., 2025), google/gemma-3-4b-it, google/gemma-3-12b-it (Team, 2025), HuggingFaceTB/SmolVLM2-2.2B-Instruct (Marafioti et al., 2025), and LiquidAI/LFM2.5-VL-1.6B (Amini et al., 2025). We use the same frozen backbones for both the text-based baselines and Vision Wormhole, ensuring that differences come from the communication protocol rather than model fine-tuning. Our main experiments focus on heterogeneous settings with small backbones, including both two-backbone configurations and a four-backbone pool; Table 1 summarizes these model combinations and role assignments. We also evaluate a 4B–12B two-backbone setting in a separate appendix section (Appendix E, Table 6). In addition, we report a *weakly supervised* codec variant trained with fewer than 100 anchor texts on a subset of the main two-backbone configurations (Appendix B.1).

MAS Protocols. We focus on sequential role-based collaboration following Zou et al. (2025), where agents follow a fixed workflow (e.g., Planner → Critic → Refiner → Judger) and exchange messages between steps. For TextMAS, messages are exchanged as natural language. For Vision Wormhole, the sender's reasoning traces are encoded by a compact universal codec into a fixed-size visual token space and injected through the receiver's vision pathway, yielding bounded message bandwidth and a fixed latent-step budget.

Codec Variants. Unless otherwise specified, we use our standard codec training recipe . We additionally evaluate a weakly supervised variant that learns the codec using fewer than 100 anchor texts on a subset of the main two-backbone configurations. Appendix B.1 provides the codec training setup for both settings.

Table 1: **Main heterogeneous MAS model configurations.** For two-backbone settings, four roles alternate between two backbones across a four-step workflow (Planner → Critic → Refiner → Judger).

Backbone Setup	Backbones	Role assignment (Planner, Critic, Refiner, Judger)
Two-backbone configurations		
2 backbones	Gemma-3-4B + SmolVLM2-2.2B	(SmolVLM2, Gemma-3-4B, SmolVLM2, Gemma-3-4B)
2 backbones	Qwen3-VL-2B + SmolVLM2-2.2B	(SmolVLM2, Qwen3-VL-2B, SmolVLM2, Qwen3-VL-2B)
2 backbones	Qwen3-VL-2B + Gemma-3-4B	(Gemma-3-4B, Qwen3-VL-2B, Gemma-3-4B, Qwen3-VL-2B)
2 backbones	LFM2.5-VL-1.6B + Gemma-3-4B	(LFM2.5-VL-1.6B, Gemma-3-4B, LFM2.5-VL-1.6B, Gemma-3-4B)
2 backbones	LFM2.5-VL-1.6B + Qwen3-VL-2B	(LFM2.5-VL-1.6B, Qwen3-VL-2B, LFM2.5-VL-1.6B, Qwen3-VL-2B)
Four-backbone pool (1.6B–4B)		
4 backbones	SmolVLM2-2.2B + LFM2.5-VL-1.6B + Gemma-3-4B + Qwen3-VL-2B	(SmolVLM2, LFM2.5-VL-1.6B, Gemma-3-4B, Qwen3-VL-2B)

Table 2: **Main heterogeneous MAS results across datasets.** Each cell reports accuracy (%) and average wall-clock time (s/query, batch-normalized); Improv. reports ΔAcc (pp) and speedup (×) of VW vs Text.

	P/R: Gemma-3-4B C/J: Qwen3-VL-2B			P/R: LFM2.5-VL-1.6B C/J: Gemma-3-4B			P/R: LFM2.5-VL-1.6B C/J: Qwen3-VL-2B			P/R: SmolVLM2-2.2B C/J: Gemma-3-4B			P/R: SmolVLM2-2.2B C/J: Qwen3-VL-2B			P: SmolVLM2-2.2B, C: LFM2.5-VL-1.6B R: Gemma-3-4B, J: Qwen3-VL-2B		
Dataset	**Text**	**VW**	**Improv.**	**Text**	**VW**	**Improv.**	**Text**	**VW**	**Improv.**	**Text**	**VW**	**Improv.**	**Text**	**VW**	**Improv.**	**Text**	**VW**	**Improv.**
GSM8K	80.8% (27.3s)	76.2% (26.7s)	-4.6pp 1.02×	71.7% (14.3s)	85.1% (14.9s)	+13.4pp 0.96×	70.9% (40.2s)	76.6% (23.0s)	+5.7pp 1.75×	67.8% (22.3s)	85.4% (11.4s)	+17.6pp 1.96×	64.3% (63.8s)	74.8% (33.5s)	+10.5pp 1.90×	62.8% (37.9s)	75.7% (26.3s)	+12.9pp 1.44×
ARC-Easy	93.4% (33.1s)	92.4% (22.0s)	-1.0pp 1.50×	88.6% (14.7s)	90.8% (9.3s)	+2.2pp 1.58×	91.4% (38.9s)	91.7% (30.1s)	+0.3pp 1.29×	84.4% (24.5s)	90.2% (7.3s)	+5.8pp 3.36×	88.6% (51.1s)	92.3% (28.2s)	+3.7pp 1.81×	85.0% (36.7s)	92.0% (21.4s)	+7.0pp 1.71×
ARC-Challenge	86.0% (49.0s)	82.1% (29.5s)	-3.9pp 1.66×	77.0% (16.0s)	81.1% (10.5s)	+4.1pp 1.52×	81.7% (46.1s)	81.7% (38.2s)	+0.0pp 1.21×	70.6% (26.0s)	80.7% (8.4s)	+10.1pp 3.10×	78.2% (68.5s)	81.7% (38.2s)	+3.5pp 1.79×	74.3% (43.2s)	81.2% (28.1s)	+6.9pp 1.54×
GPQA	29.8% (348.4s)	39.9% (174.6s)	+10.1pp 2.00×	31.3% (65.4s)	24.2% (47.2s)	-7.1pp 1.39×	42.4% (403.7s)	34.9% (221.7s)	-7.5pp 1.82×	26.3% (101.4s)	29.3% (42.1s)	+3.0pp 2.41×	32.3% (483.3s)	37.9% (225.5s)	+5.6pp 2.14×	33.8% (315.4s)	36.9% (168.6s)	+3.1pp 1.87×
MedQA	53.3% (91.5s)	48.0% (83.0s)	-5.3pp 1.10×	47.7% (25.9s)	52.3% (17.9s)	+4.6pp 1.45×	51.3% (109.5s)	49.7% (104.3s)	-1.6pp 1.05×	41.0% (34.0s)	48.3% (13.8s)	+7.3pp 2.46×	44.7% (125.0s)	47.0% (93.4s)	+2.3pp 1.34×	46.3% (101.2s)	47.7% (80.7s)	+1.4pp 1.25×
MBPP-Plus	50.5% (108.7s)	51.3% (69.2s)	+0.8pp 1.57×	45.8% (8.9s)	66.4% (11.3s)	+20.6pp 0.79×	45.0% (80.6s)	51.1% (86.7s)	+6.1pp 0.93×	44.7% (17.5s)	67.7% (8.0s)	+23.0pp 2.19×	37.8% (125.6s)	47.9% (79.9s)	+10.1pp 1.57×	28.8% (43.7s)	47.9% (68.0s)	+19.1pp 0.64×
HumanEval-Plus	40.9% (121.6s)	37.2% (80.1s)	-3.7pp 1.52×	43.9% (11.3s)	60.4% (19.8s)	+16.5pp 0.57×	38.4% (86.0s)	37.8% (101.1s)	-0.6pp 0.85×	32.9% (25.1s)	59.1% (15.3s)	+26.2pp 1.64×	31.1% (126.1s)	40.9% (100.3s)	+9.8pp 1.26×	19.5% (46.3s)	37.8% (79.2s)	+18.3pp 0.58×
AIME 2024	23.3% (1314.4s)	36.7% (385.8s)	+13.4pp 3.41×	0.0% (267.5s)	6.7% (90.9s)	+6.7pp 2.94×	30.0% (2104.5s)	20.0% (541.6s)	-10.0pp 3.89×	3.3% (120.2s)	6.7% (61.8s)	+3.4pp 1.94×	13.3% (2806.9s)	23.3% (513.3s)	+10.0pp 5.47×	13.3% (1234.8s)	26.7% (415.7s)	+13.4pp 2.97×
AIME 2025	16.7% (1432.9s)	26.7% (382.0s)	+10.0pp 3.75×	3.3% (127.7s)	13.3% (81.8s)	+10.0pp 1.56×	13.3% (1444.9s)	20.0% (501.3s)	+6.7pp 2.88×	3.3% (149.4s)	10.0% (59.8s)	+6.7pp 2.50×	16.7% (1996.1s)	23.3% (505.7s)	+6.6pp 3.95×	26.7% (777.2s)	20.0% (395.3s)	-6.7pp 1.97×

Baselines. Our primary comparison is against standard text-mediated MAS (TextMAS) under identical agent roles and prompts. We also acknowledge recent latent communication methods that could, in principle, enable heterogeneous latent collaboration, including Cache-to-Cache (Fu et al., 2025) and LatentMAS-style latent collaboration (and hybrid variants) (Zou et al., 2025). In our model configurations, our attempts to adapt these methods to our heterogeneous backbones produced degenerate and unstable generations, preventing a reliable quantitative comparison; we therefore focus on TextMAS as the main baseline and leave a thorough evaluation against these latent baselines to future work.

Implementation Details. We provide hardware placement, generation budgets, and dynamic batching details in Appendix B.2.

4.2 Main Results

We report system-level accuracy and end-to-end wall-clock time per query (batch-normalized; seconds/query) for TextMAS and Vision Wormhole (VW) across heterogeneous model configurations. For the main settings we use a compact cell format (accuracy and time per cell); for all settings we report improvements as ΔAcc (percentage points) and speedup (×) of VW relative to TextMAS.

4.2.1 Main Heterogeneous MAS

Table 2 shows that VW consistently reduces end-to-end wall-clock time while often improving accuracy in the main regime. Macro-averaged across all reported dataset–configuration entries in this table, VW improves accuracy by +6.3pp and speeds up inference by 1.87×. Gains are strongest on code generation (*MBPP-Plus* and *HumanEval-Plus*), with an average +13.2pp improvement, while still providing a 1.21× speedup. We also observe clear configuration sensitivity: certain role assignments yield speedups with small or negative ΔAcc on the hardest benchmarks, suggesting that the optimal communication interface depends on the sender/receiver pairing and which backbone occupies planning vs. critique roles.

Table 3: **Weakly supervised codec results across datasets.** We report accuracy (%), wall-clock time (s/query, batch-normalized), and improvement (ΔAcc in pp, speedup in ×) of VW vs Text.

	P/R: Gemma-3-4B, C/J: Qwen3-VL-2B						P/R: SmolVLM2-2.2B, C/J: Qwen3-VL-2B					
Dataset	**Text Acc**	**Text Time**	**VW Acc**	**VW Time**	**ΔAcc**	**Speedup**	**Text Acc**	**Text Time**	**VW Acc**	**VW Time**	**ΔAcc**	**Speedup**
GSM8K	80.8%	27.3s	77.6%	22.9s	-3.2pp	1.19×	64.3%	63.8s	77.0%	25.6s	+12.7pp	2.49×
ARC-Easy	93.4%	33.1s	90.3%	23.1s	-3.1pp	1.43×	88.6%	51.1s	91.8%	23.6s	+3.2pp	2.17×
ARC-Challenge	86.0%	49.0s	80.5%	30.3s	-5.5pp	1.62×	78.2%	68.5s	81.3%	30.4s	+3.1pp	2.25×
GPQA	29.8%	348.4s	34.9%	172.9s	+5.1pp	2.02×	32.3%	483.3s	33.3%	172.5s	+1.0pp	2.80×
MedQA	53.3%	91.5s	45.0%	82.0s	-8.3pp	1.12×	44.7%	125.0s	49.0%	75.8s	+4.3pp	1.65×
MBPP-Plus	50.5%	108.7s	46.6%	69.1s	-3.9pp	1.57×	37.8%	125.6s	49.2%	69.4s	+11.4pp	1.81×
HumanEval-Plus	40.9%	121.6s	40.2%	80.7s	-0.7pp	1.51×	31.1%	126.1s	42.7%	80.6s	+11.6pp	1.56×
AIME 2024	23.3%	1314.4s	26.7%	404.8s	+3.4pp	3.25×	13.3%	2806.9s	36.7%	389.8s	+23.4pp	7.20×
AIME 2025	16.7%	1432.9s	23.3%	370.5s	+6.6pp	3.87×	16.7%	1996.1s	23.3%	415.8s	+6.6pp	4.80×

Table 4: **Single-agent baseline vs combined heterogeneous MAS performance.** For each model, *Combined* TextMAS/VW reports the per-dataset *mean* accuracy over all heterogeneous MAS configurations that include that model. Averages are macro means over shared datasets. Δ columns are Combined minus Single-agent baseline (percentage points).

Model	**MAS Cfgs**	**Datasets**	**Single**	**TextMAS**	**VW**	**ΔText**	**ΔVW**
Qwen3-VL-2B	6	9	50.8	49.4	52.6	-1.4	+1.8
Gemma-3-4B	6	9	55.7	52.5	56.0	-3.2	+0.3
SmolVLM2-2.2B	4	9	25.8	44.2	52.7	+18.5	+26.9
LFM2.5-VL-1.6B	3	9	40.2	46.8	52.2	+6.6	+12.0
Macro Avg.	–	–	43.1	48.2	53.4	+5.1	+10.3

Preliminary mid-sized (4B–12B) results and analysis are reported in Appendix E.

4.3 Weakly Supervised Codec Variant

The main results above assume our default codec training recipe, which uses a small but non-trivial number of anchor texts to supervise latent-to-vision injection. We next stress-test whether the communication channel remains effective under *weak supervision*.

Concretely, we train codecs using fewer than 100 anchor texts (Appendix B.1) and evaluate the same heterogeneous collaboration protocols without changing the frozen backbones, prompts, or decoding budgets. Table 3 summarizes the resulting performance. Across the reported entries, VW continues to provide large runtime reductions and typically improves task accuracy relative to TextMAS, with an average gain of +6.5pp and a 2.67× speedup. These results support our central hypothesis that the VLM visual interface functions as a robust continuous port for inter-agent communication, enabling practical codec training even when only a handful of anchor texts are available.

4.4 Best Single-Agent Baseline vs Combined MAS

Prior work has observed that multi-agent LLM systems may underperform the best single model for reasons beyond communication bandwidth, including coordination and aggregation failures, correlated errors, and routing mistakes (Pappu et al., 2026; Zhang et al., 2025a). Accordingly, our evaluation uses two complementary views. First, the main and weakly supervised results provide a controlled within-MAS comparison (TextMAS vs. VW) under matched prompts, role assignments, numbers of agents/backbones, interaction protocol, and decoding budgets, so the communication mechanism is the only changed factor. Second, we compare each model's single-agent baseline against its *combined* MAS performance, where "combined" means averaging over all heterogeneous configurations that include that model, to contextualize orchestration effects.

Two patterns are consistent. First, for stronger backbones (Qwen3-VL-2B and Gemma-3-4B), TextMAS shows noticeable drops relative to single-agent baselines, while Vision Wormhole

(VW) stays much closer to parity (and slightly above parity in this table). Second, for weaker backbones (SmolVLM2-2.2B and LFM2.5-VL-1.6B), both MAS variants improve over single-agent performance, with larger gains under VW.

Taken together, this comparison indicates that VW is more robust to heterogeneous orchestration effects. In mixed-capability teams, VW stays closer to strong-model single-agent performance than TextMAS while still preserving the collaborative gains observed for weaker backbones, consistent with bounded latent communication reducing cross-role interference.

5 Conclusion

We introduced the *Vision Wormhole*, a latent communication framework that repurposes the visual interface of VLMs as a universal, continuous port for heterogeneous multi-agent collaboration. By translating sender-side reasoning traces into a fixed-size vision-token message via a lightweight Universal Visual Codec, our approach provides a *bounded* and *modular* alternative to text communication and pairwise cache translators, reducing multi-family integration from quadratic to linear scaling in the number of participating models. Empirically, Vision Wormhole yields consistent end-to-end speedups in heterogeneous settings and, in the lightweight regime, often improves accuracy while reducing end-to-end wall-clock time. We also find that a weakly supervised codec trained from fewer than 100 anchor texts can still deliver meaningful gains, suggesting that the vision pathway is a data-efficient channel for latent transfer.

Acknowledgements

We thank Contextual AI for supporting the experiments in this paper.

References

Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey. *IEEe Access*, 2025.

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Jean-Baptiste Alayrac et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

Alexander Amini, Anna Banaszak, Harold Benoit, Arthur Böök, Tarek Dakhran, Song Duong, Alfred Eng, Fernando Fernandes, Marc Härkönen, Anne Harrington, Ramin Hasani, Saniya Karwa, Yuri Khrustalev, Maxime Labonne, Mathias Lechner, Valentine Lechner, Simon Lee, Zetian Li, Noel Loo, Jacob Marks, Edoardo Mosca, Samuel J. Paech, Paul Pak, Rom N. Parnichkun, Alex Quach, Ryan Rogers, Daniela Rus, Nayan Saxena, Bettina Schlager, Tim Seyde, Jimmy T. H. Smith, Aditya Tadimeti, and Neehal Tumma. Lfm2 technical report, 2025. URL https://arxiv.org/abs/2511.23404.

Raviteja Anantha, Bortik Bandyopadhyay, Anirudh Kashi, Sayantan Mahinder, Andrew W Hill, and Srinivas Chappidi. Protip: Progressive tool retrieval improves planning, 2023. URL https://arxiv.org/abs/2312.10332.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang,

Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Yash Bansal et al. Revisiting model stitching to compare neural representations. In *NeurIPS*, 2021.

Timo Birr, Christoph Pohl, Abdelrahman Younes, and Tamim Asfour. Autogpt+p: Affordance-based task planning with large language models, 2024. URL https://arxiv.org/abs/2402.10778.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. *arXiv preprint arXiv:2410.08115*, 2024.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11534–11557, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018a.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Julian Coda-Forno, Zhuokai Zhao, Qiang Zhang, Dipesh Tamboli, Weiwei Li, Xiangjun Fan, Lizhu Zhang, Eric Schulz, and Hsiao-Ping Tseng. Exploring system 1 and 2 communication for latent reasoning in llms. *arXiv preprint arXiv:2510.00494*, 2025.

CM Downey, Terra Blevins, N Goldfine, et al. Embedding structure matters: Comparing methods to adapt multilingual vocabularies to new languages. *arXiv preprint arXiv:2307.01423*, 2023.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv preprint arXiv:2505.00675*, 2025.

Albert Einstein and Nathan Rosen. The particle problem in the general theory of relativity. *Physical Review*, 48(1):73, 1935.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. Retrofitting large language models with dynamic tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Benjamin Fein-Ashley and Jacob Fein-Ashley. Bridging hidden states in vision-language models. *arXiv preprint arXiv:2511.11526*, 2025.

Zhaohan Feng, Ruiqi Xue, Lei Yuan, Yang Yu, Ning Ding, Meiqin Liu, Bingzhao Gao, Jian Sun, Xinhu Zheng, and Gang Wang. Multi-agent embodied ai: Advances and future directions. *arXiv preprint arXiv:2505.05108*, 2025.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. *arXiv preprint arXiv:2510.03215*, 2025.

C Goddard and FF Neto. Training-free tokenizer transplantation via orthogonal matching pursuit. *arXiv preprint arXiv:2506.06607*, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2023.

Zhipeng Hou, Junyi Tang, and Yipeng Wang. Halo: Hierarchical autonomous logic-oriented orchestration for multi-agent llm systems. *arXiv preprint arXiv:2505.13516*, 2025.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025a.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025b.

Chao Jia et al. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning. *arXiv preprint arXiv:2509.24251*, 2025a.

Chao Li, Jinchao Zhang, and Chengqing Zong. Tokalign: Efficient vocabulary adaptation via token alignment. *arXiv preprint arXiv:2506.03523*, 2025b.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: communicative agents for "mind" exploration of large language model society. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023a. Curran Associates Inc.

Junnan Li et al. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023b.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Zhuo Li, Weiran Wu, Yunlong Guo, Jian Sun, and Qing-Long Han. Embodied multi-agent systems: A review. *IEEE/CAA Journal of Automatica Sinica*, 12(6):1095–1116, 2025c.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025d.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency, 2023a. URL https://arxiv.org/abs/2304.11477.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023b.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation. *arXiv preprint arXiv:2412.17747*, 2024.

Xiaoze Liu, Weichen Yu, Matt Fredrikson, Xiaoqian Wang, and Jing Gao. The trojan in the vocabulary: Stealthy sabotage of llm composition, 2026. URL https://arxiv.org/abs/2601.00065.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models, 2025. URL https://arxiv.org/abs/2504.05299.

math ai. AIME 2025 dataset. https://huggingface.co/datasets/math-ai/aime25, 2025.

Maxwell-Jia. AIME 2024 dataset. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024.

Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. *arXiv preprint arXiv:2503.20083*, 2025.

Luca Moroni, Giovanni Puccetti, Pere-Lluís Huguet Cabot, et al. Optimizing llms for italian: Reducing token fertility and enhancing efficiency through vocabulary adaptation. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

N Mundra, ANK Khandavally, R Dabre, et al. An empirical comparison of vocabulary expansion and initialization approaches for language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 2024.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, et al. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

Aneesh Pappu, Batu El, Hancheng Cao, Carmelo di Nolfo, Yanchao Sun, Meng Cao, and James Zou. Multi-agent teams hold experts back, 2026. URL https://arxiv.org/abs/2602.01011.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023a.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023b.

Pouya Pezeshkpour, Eser Kandogan, Nikita Bhutani, Sajjadur Rahman, Tom Mitchell, and Estevam Hruschka. Reasoning capacity in multi-agent systems: Limitations, challenges and human-centered solutions. *arXiv preprint arXiv:2402.01108*, 2024.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Unks everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 10186–10203, 2021.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, et al. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*, 2025.

Alec Radford et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

François Remy, Pieter Delobelle, Bettina Berendt, Kris Demuynck, et al. Tik-to-tok: Translating language models one token at a time: An embedding initialization strategy for efficient language adaptation. *arXiv preprint arXiv:2308.16898*, 2023.

François Remy, Pieter Delobelle, Hayk Avetisyan, et al. Trans-tokenization and cross-lingual vocabulary transfers: Language adaptation of llms for low-resource nlp. *arXiv preprint arXiv:2408.04303*, 2024.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, et al. Tptu: large language model-based ai agents for task planning and tool usage. *arXiv preprint arXiv:2308.03427*, 2023.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

S Sharthak, V Pahalwan, A Kamath, et al. Achieving tokenizer flexibility in language models through heuristic adaptation and supertoken learning. *arXiv preprint arXiv:2505.09738*, 2025.

Yuling Shi, Chaoxiang Xie, Zhensu Sun, Yeheng Chen, Chenxu Zhang, Longfei Yun, Chengcheng Wan, Hongyu Zhang, David Lo, and Xiaodong Gu. Codeocr: On the effectiveness of vision language models in code understanding, 2026. URL https://arxiv.org/abs/2602.01785.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Wen Tai, H.T. Kung, Xin Luna Dong, Marcus Comiter, and C.-F. Kuo. exbert: Extending pre-trained models with domain-specific vocabulary under constrained training resources. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 2020.

Wei Tao, Yucheng Zhou, Yanlin Wang, Wenqiang Zhang, Hongyu Zhang, and Yu Cheng. Magis: Llm-based multi-agent framework for github issue resolution. *Advances in Neural Information Processing Systems*, 37:51963–51993, 2024.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

Giorgos Vernikos and Andrei Popescu-Belis. Subword mapping and anchoring across languages. *arXiv preprint arXiv:2109.04556*, 2021.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024a.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language. *arXiv preprint arXiv:2511.21395*, 2025a.

Rui Wang, Hongru Wang, Boyang Xue, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai Wong, Heng Ji, and Kam-Fai Wong. Harnessing the reasoning economy: A survey of efficient reasoning for large language models. *arXiv preprint arXiv:2503.24377*, 2025b.

Yifan Wang, Shiyu Li, Peiming Li, Xiaochen Yang, Yang Tang, and Zheng Wei. Render-of-thought: Rendering textual chain-of-thought as images for visual latent reasoning, 2026. URL https://arxiv.org/abs/2601.14750.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 50453–50466, 2024b.

Zhao Wang, Sota Moriyama, Wei-Yao Wang, Briti Gangopadhyay, and Shingo Takamatsu. Talk structurally, act hierarchically: A collaborative framework for llm multi-agent systems. *arXiv preprint arXiv:2502.11098*, 2025c.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression, 2025. URL https://arxiv.org/abs/2510.18234.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Feijie Wu, Zitao Li, Fei Wei, Yaliang Li, Bolin Ding, and Jing Gao. Talk to right specialists: Routing and planning in multi-agent system for question answering. *arXiv preprint arXiv:2501.07813*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. An empirical study on cross-lingual vocabulary adaptation for efficient language model inference. *arXiv preprint arXiv:2402.10712*, 2024.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. How can we effectively expand the vocabulary of llms with 0.01 gb of target language text? *Computational Linguistics*, 2025.

Bingyu Yan, Zhibo Zhou, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, Zhoujun Li, Chaozhuo Li, and Xiaoming Zhang. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *arXiv preprint arXiv:2502.14321*, 2025.

Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. Llm-medqa: Enhancing medical question answering through case studies in large language models. *arXiv preprint arXiv:2501.05464*, 2024a.

Yingxuan Yang, Qiuying Peng, Jun Wang, Ying Wen, and Weinan Zhang. Llm-based multi-agent systems: Techniques and business perspectives. *arXiv preprint arXiv:2411.14033*, 2024b.

Huanjin Yao, Ruifei Zhang, Jiaxing Huang, Jingyi Zhang, Yibo Wang, Bo Fang, Ruolin Zhu, Yongcheng Jing, Shunyu Liu, Guanbin Li, et al. A survey on agentic multimodal large language models. *arXiv preprint arXiv:2510.10991*, 2025.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, et al. Kvcomm: Online cross-context kv-cache communication for efficient llm-based multi-agent systems. *arXiv preprint arXiv:2510.12872*, 2025.

Xiaohua Zhai et al. Sigmoid loss for language image pre-training. In *ICCV*, 2023.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, et al. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024a.

Hangfan Zhang, Zhiyao Cui, Jianhao Chen, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. Stop overvaluing multi-agent debate – we must rethink evaluation and embrace model heterogeneity, 2025a. URL https://arxiv.org/abs/2502.08788.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025b.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024b.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025c.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025d.

Bingxi Zhao, Lin Geng Foo, Ping Hu, Christian Theobalt, Hossein Rahmani, and Jun Liu. Llm-based agentic reasoning frameworks: A survey from methods to scenarios. *arXiv preprint arXiv:2508.17692*, 2025a.

Jiaxing Zhao, Hongbin Xie, Yuzhen Lei, Xuan Song, Zhuoran Shi, Lianxin Li, Shuangxue Liu, and Haoran Zhang. Connecting the dots: A chain-of-collaboration prompting framework for llm agents. *arXiv preprint arXiv:2505.10936*, 2025b.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2025c.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *arXiv preprint arXiv:2510.20733*, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models, 2024. URL https://arxiv.org/abs/2310.04406.

Heng Zhou, Hejia Geng, Xiangyuan Xue, Li Kang, Yiran Qin, Zhiyong Wang, Zhenfei Yin, and Lei Bai. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. *arXiv preprint arXiv:2503.02390*, 2025.

Wenxuan Zhou, Junyi Du, and Xiang Ren. Improving bert fine-tuning with embedding normalization. *arXiv preprint arXiv:1911.03918*, 2019.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search, 2023. URL https://arxiv.org/abs/2310.13227.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

A Related Work

A.1 LLM-based Multi-Agent Systems and Communication Bottlenecks

LLM-based multi-agent systems (MAS) have rapidly expanded from conceptual overviews to practical deployments, with surveys consolidating common coordination patterns, agent roles, and evaluation practices across application domains.(Guo et al., 2024; Tran et al., 2025; Yan et al., 2025; Yang et al., 2024b; Acharya et al., 2025; Wang et al., 2024a; Yao et al., 2025; Li et al., 2025c; Feng et al., 2025; Zhang et al., 2024a; Zhao et al., 2025a) Most existing MAS instantiate collaboration as token-level interaction: agents communicate via natural-language messages (often with structured prompting), optionally with orchestration layers that manage routing, delegation, and tool calls.(Wu et al., 2024; Hong et al., 2023; Li et al., 2023a; Fourney et al., 2024; Hou et al., 2025; Hu et al., 2025b; Tao et al., 2024; Park et al., 2023a; Wu et al., 2025; Wang et al., 2025c) This design is attractive because it is model-agnostic and easy to audit, but it makes communication a dominant cost driver: token messages are slow, bandwidth-limited under context constraints, and can discard fine-grained intermediate information that would be useful for downstream reasoning.

A parallel line of work improves collaboration quality by designing multi-step workflows (e.g., chain-based collaboration and self-improvement loops) and by optimizing multi-agent efficiency under fixed compute budgets.(Zhang et al., 2024b; Zhao et al., 2025b;c; Chen et al., 2024; 2025; Zhou et al., 2025; Li et al., 2025d; Zhuge et al., 2024) Complementary studies analyze why multi-agent reasoning fails in practice, highlighting brittleness in information sharing, coordination overheads, and miscalibrated specialization.(Pezeshkpour et al., 2024; Cemri et al., 2025) Our work targets a specific, recurring bottleneck across these systems: the communication interface. Rather than proposing a new coordination policy, we focus on improving *interoperability and bandwidth* when agents come from different model families.

Finally, long-horizon MAS often rely on memory, planning, and tool-use components.(Zhang et al., 2025c; Du et al., 2025; Packer et al., 2023; Zhong et al., 2024; Wang et al., 2024b; Hu et al., 2025a; Pan et al., 2025; Ruan et al., 2023; Zhou et al., 2024; Liu et al., 2023a; Birr et al., 2024; Anantha et al., 2023; Zhuang et al., 2023) These modules mainly address *what* agents do and *how* they act; our contribution is orthogonal, addressing *how* heterogeneous agents can exchange information efficiently.

A.2 Latent-Space Communication for Multi-Agent Collaboration

To reduce the verbosity of token-based interaction, recent work explores collaboration directly in representation space. One direction replaces text messages with *latent messages* (e.g., hidden states or KV-caches), enabling faster exchange and preserving richer intermediate signals.(Zou et al., 2025; Fu et al., 2025; Ye et al., 2025; Zheng et al., 2025) A key distinction is whether the approach assumes *homogeneous* agents or supports *heterogeneous* model families.

Training-free latent exchange within homogeneous settings. Several systems enable tokenless or cache-level interaction without additional training by reusing intermediate activations produced by a shared backbone.(Zou et al., 2025; Ye et al., 2025) These approaches can substantially reduce communication overhead, but they typically rely on compatibility of internal representations (e.g., comparable layer structure, hidden dimensionality, or cache semantics), making cross-family interoperability challenging.

Learned bridges for cross-model latent transfer. Another line of work trains translation or fusion modules that map one model's internal states into another's latent space.(Fu et al., 2025; Zheng et al., 2025) This can enable cross-family latent transfer, but learned bridges introduce additional supervision requirements and engineering complexity, and naïvely scale poorly as the number of agent families grows because pairwise bridges can induce $\mathcal{O}(N^2)$ training and maintenance.

A.3 Latent Reasoning and Continuous Thought

In parallel to latent communication, a growing literature studies *latent reasoning* to reduce token-level chain-of-thought verbosity and improve efficiency.(Hao et al., 2024; Zhang et al., 2025d; Zhu et al., 2025; Liu et al., 2024; Coda-Forno et al., 2025; Qu et al., 2025; Sui et al., 2025; Wang et al., 2025b; Zhang et al., 2025b) These methods typically focus on how a single model can internalize intermediate computations in continuous space (e.g., replacing explicit textual rationales with continuous representations or augmenting the cache with differentiable deliberation steps).(Hao et al., 2024; Liu et al., 2024) Our work is complementary: we leverage the same motivation (tokens as a bandwidth bottleneck) but in a multi-agent setting, where the core challenge becomes *inter-model interoperability* rather than only intra-model efficiency.

A.4 Interoperability Across Model Families: Tokenizers, Representations, and Multimodal Anchors

Heterogeneity is a central obstacle to collaboration across independently trained LLM/VLM families. Tokenization mismatch has been addressed via tokenizer transfer, dynamic tokenization, cross-tokenizer distillation, and vocabulary alignment/expansion strategies.(Minixhofer et al., 2024; Feher et al., 2025; Minixhofer et al., 2025; Li et al., 2025b; Remy et al., 2023; 2024; Tai et al., 2020; Pfeiffer et al., 2021; Rust et al., 2021; Vernikos & Popescu-Belis, 2021; Mundra et al., 2024; Yamaguchi et al., 2024; 2025; Moroni et al., 2025; Downey et al., 2023; Goddard & Neto, 2025; Sharthak et al., 2025; Liu et al., 2026) While these techniques improve transfer or inference efficiency, they primarily operate by modifying tokenization and embeddings, and do not directly provide a shared *latent* communication substrate for MAS.

Beyond tokenizers, representation-level alignment and model compatibility have been studied through linear/affine correspondences, model stitching, and permutation-aware merging, motivating when simple mappings can relate internal features across networks.(Bansal et al., 2021; Lenc & Vedaldi, 2015; Ainsworth et al., 2022; Wortsman et al., 2022; Park et al.,

2023b; Zou et al., 2023; Zhou et al., 2019) These insights inform our choice of lightweight affine alignment, but prior work is not tailored to multi-agent communication nor does it provide a modality-grounded shared interface.

Finally, multimodal pretraining demonstrates that vision can serve as a strong semantic anchor for aligning representations across modalities and architectures.(Radford et al., 2021; Jia et al., 2021; Zhai et al., 2023; Alayrac et al., 2022; Li et al., 2023b) Recent efforts also explore aligning modality-specific hidden states inside vision-language models and performing reasoning directly in latent visual spaces.(Fein-Ashley & Fein-Ashley, 2025; Li et al., 2025a; Wang et al., 2025a) Our approach operationalizes this idea for multi-agent interoperability: by exploiting visual input, we construct a shared codec space that is decoupled from any single model's tokenizer and hidden-state idiosyncrasies, enabling modular cross-family latent communication with minimal per-family adaptation.

A.5 Positioning of Our Work

Our work is best viewed as a *communication-interface* contribution for heterogeneous MAS, rather than a new coordination policy. Relative to text-mediated MAS, we keep the same role workflow but replace token exchange with bounded latent communication through the VLM visual interface. Relative to homogeneous latent sharing, we target cross-family interoperability, where hidden spaces and tokenizers are mismatched by design. Relative to learned pairwise translators, we use a modular hub-and-spoke formulation with per-family codec/alignment components, reducing maintenance from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ as families scale. This places Vision Wormhole at the intersection of efficient communication, heterogeneous compatibility, and practical extensibility for real-world multi-model agent systems.

B Implementation Details

B.1 Codec Training Setup (Shared Across Runs)

Objective. We train a latent-to-vision injection codec for each backbone model. At inference time, we can optionally *merge* codecs across models to enable multi-agent communication without retraining from scratch.

Anchor corpora. We consider two anchor settings: (i) the **default** setting uses 3,000 total examples (1,000 each from `cos_e`, `OpenCodeReasoning`, and `PRM800K`); and (ii) a **weakly supervised** setting uses 90 total examples (30 each from the same three sources).

Anchor construction. For each data source, we cap the number of sampled examples, concatenate sources, and shuffle with a fixed random seed. We enable streaming dataset loading to avoid full materialization when constructing anchors.

Step-based random sampling. Training uses step-based random sampling rather than strict epoch passes: at each optimization step we sample a mini-batch uniformly at random from the anchor pool (batch size 2). With 400 optimization steps, this yields 800 anchor draws per model. This corresponds to an effective exposure of $\approx 0.27\times$ dataset coverage for the default 3,000-anchor setting (800/3000), and $\approx 8.9\times$ for the weakly supervised 90-anchor setting (800/90).

Codec architecture hyperparameters. Unless otherwise stated, we set the universal token dimension to $D = 512$, the number of codec tokens to $K_{\text{u}} = 1024$, and the number of image-side injection tokens to $K_{\text{img}} = 256$. The codec uses 6 transformer layers with 8 attention heads and dropout 0.10. The latent rollout length is fixed to $T = 1024$ steps.

Optimization and losses. We optimize with AdamW at a learning rate of 2×10^{-4} for 400 steps with batch size 2. The training objective is a weighted sum of three terms:

- Hidden-state MSE (weight 1.0).

Table 5: **Per-dataset generation budgets and default batch sizes.** We adopt the same `max_new_tokens` settings as (Zou et al., 2025).

Dataset	`max_new_tokens`	Default batch size
GSM8K	2048	12
ARC-Easy	2048	12
ARC-Challenge	2048	12
MedQA	4096	8
MBPP-Plus	4096	8
HumanEval-Plus	4096	8
GPQA	8192	4
AIME 2024	20000	4
AIME 2025	20000	4

- Logit alignment via a KL-style divergence (weight 0.25; temperature 1.0).
- Injection-statistics regularizer (weight 0.1).

We apply gradient clipping with max-norm 1.0 and use standard numerical stabilization, including clipping for latent/logit/injection values and non-finite guards.

Rollout mode and alignment placeholders. By default, codec training uses latent-space rollout with a single Monte Carlo rollout (effectively deterministic under our settings). Single-model codec training skips expensive cross-model alignment; we use identity mappings for alignment placeholders when needed.

What changes across variants. Across codec variants, we vary the backbone model, the anchor corpus size (default vs. weakly supervised), and the merge pairing used at inference. We keep the core codec architecture, optimizer/loss weights, step count, batch size, and the step-based sampling strategy fixed.

Merged codecs. For multi-model inference, we merge per-model codec checkpoints and refit universal-space alignment with a closed-form ridge regression, rather than retraining a new multi-model codec end-to-end.

B.2 Experiment Runtime and Generation Budgets

Decoding and evaluation. Unless otherwise stated, we use greedy decoding for evaluation and keep prompts and generation budgets consistent across methods (TextMAS vs. Vision Wormhole) within each task.

Hardware and model placement. All experiments are conducted on NVIDIA A6000 GPUs. For two-backbone runs in the main setting, we colocate both backbones on a single GPU. For the four-backbone pool and for mid-sized two-backbone runs (4B–12B), we use two GPUs and place half of the agents on each GPU.

Per-dataset token budgets. Following LatentMAS (Zou et al., 2025), we set the maximum number of newly generated tokens (`max_new_tokens`) per dataset, shared by both TextMAS and Vision Wormhole.

Dynamic batching and time reporting. We choose the default generation batch size based on the token budget: `max_new_tokens` $\leq 2048 \Rightarrow 12$, $\leq 4096 \Rightarrow 8$, and $> 4096 \Rightarrow 4$. To improve robustness, we adopt a retry-based strategy for out-of-memory (OOM) failures: upon OOM, we retry with batch sizes in descending order $\{12, 8, 4, 2, 1\}$ until the run succeeds. We report average end-to-end wall-clock time per query under the above placement and batching strategy to reflect system-level communication overheads. In our implementation, per-query time is computed by normalizing batch wall-clock time by the number of samples in the batch (i.e., `batch_time / batch_samples`), so it should be interpreted as batch-normalized runtime (seconds/query) rather than interactive tail latency.

C Additional Details: Codec Training, Alignment, and Inference

This appendix provides architectural and procedural details omitted from the main text for clarity.

C.1 Training a Codec for a Specific VLM

NormMatch: keeping pseudo-tokens on the embedding-norm manifold. Latent rollouts feed back continuous pseudo-token embeddings derived from hidden states. A practical issue is that hidden-state norms may drift relative to the distribution of true token embeddings, which can destabilize the autoregressive continuation in embedding space. We therefore define a simple per-model normalization operator:

$$\mathrm{NormMatch}_i(h) \;=\; \alpha_i \cdot \frac{h}{\|h\|_2 + \epsilon}, \tag{9}$$

where α_i is the typical token-embedding norm for model i, e.g., $\alpha_i = \mathbb{E}_{w\sim\mathcal{V}_i}\|E_i(w)\|_2$, and ϵ is a small constant. This ensures the pseudo-token lives in the same norm range as embeddings observed during training.

Latent rollout with cached context. Let the prompt produce cached attention keys/values (equivalently, a fixed conditioning context). A rollout step appends a single pseudo-token embedding x_t and computes the next hidden state at the new position. Repeating for T steps yields $H_i \in \mathbb{R}^{T\times d_i}$. The rollout length T is a fixed hyperparameter that bounds message extraction cost.

Perceiver-style resampler encoder. The encoder $\mathcal{E}_i$ maps H_i to K_u tokens of dimension D. We first project the rollout into the universal dimension:

$$Z = H_i P_i \in \mathbb{R}^{T\times D}, \tag{10}$$

for a learned matrix $P_i \in \mathbb{R}^{d_i\times D}$. We then maintain a small set of learned queries $Q^{(0)} \in \mathbb{R}^{K_\mathrm{u}\times D}$ and update them through L cross-attention blocks:

$$Q^{(\ell+1)} = Q^{(\ell)} + \mathrm{MHA}\Big(\mathrm{LN}(Q^{(\ell)}), \mathrm{LN}(Z), \mathrm{LN}(Z)\Big), \tag{11}$$

$$Q^{(\ell+1)} = Q^{(\ell+1)} + \mathrm{FFN}\Big(\mathrm{LN}(Q^{(\ell+1)})\Big), \qquad \ell = 0,\dots,L-1. \tag{12}$$

This is Perceiver-style resampling: a constant number of queries attends to a variable-length latent sequence.

Global and style tokens. In addition to K semantic tokens, we include a global token (for pooling) and a style token (for scale/statistics cues). A simple and effective statistic vector is

$$s(H_i) = \left[\mathrm{mean}(H_i),\ \mathrm{std}(H_i),\ \frac{1}{T}\sum_{t=1}^{T}\|H_{i,t}\|_2\right] \in \mathbb{R}^3, \tag{13}$$

which is mapped by a small MLP into $\mathbb{R}^D$ and added to the style token. This helps stabilize cross-prompt and cross-role transfer by communicating coarse distributional properties of the rollout.

Universal-to-vision decoder. The decoder $\mathcal{D}_i$ mirrors the resampler pattern. A learned set of K_img image queries attends to the (possibly concatenated) universal tokens to produce K_img vectors, which are linearly projected into $\mathbb{R}^{d_i}$ to form Δ_i. A gate $g_i \in (0,1)$ is predicted from a pooled representation of the universal tokens. The gate serves two roles: (i) it prevents over-injection when the memory is empty or low-confidence, and (ii) it allows the codec to adapt injection strength across examples.

Dummy-image baseline and length resampling. Different VLMs use different image-token lengths $L_{\text{img}}^{(i)}$. We compute once per model a baseline image-span embedding $\bar{X}_{\text{img}}^{(i)}$ using a fixed dummy image. At inference, we resample the decoded $\Delta_i \in \mathbb{R}^{K_{\text{img}} \times d_i}$ to the required span length $\text{Resample}(\Delta_i; L_{\text{img}}^{(i)}) \in \mathbb{R}^{L_{\text{img}}^{(i)} \times d_i}$ using linear interpolation along the token index. The injected span is written as in Eq. (2).

Distillation signals and where they are taken. We distill at the prompt boundary: the teacher sees the full message m in text, and the student must match the teacher's hidden state and next-token distribution at the same boundary position. This gives dense supervision without requiring any human labels beyond collecting anchor messages. The KL term in Eq. (3) is especially informative because it provides a rich gradient over the entire vocabulary distribution (not only a single target token).

C.2 Affine Alignment in Universal Space

Why alignment is needed despite shared D. Universal tokens have a shared dimensionality but may differ by a model-specific basis (rotation, scaling, and offsets) induced by independent training of $\mathcal{E}_i$. Affine alignment creates a common coordinate system that enables modular composition.

Closed-form ridge regression. Given anchor texts $\{m_j\}_{j=1}^{M}$, we obtain token matrices $U_i(m_j)$ and $U_r(m_j)$ for model i and reference r. We flatten across anchors and token positions to form $X_i \in \mathbb{R}^{(MK_{\text{u}}) \times D}$ and $Y_r \in \mathbb{R}^{(MK_{\text{u}}) \times D}$, and solve

$$\min_{W,b} \quad \|X_i W + \mathbf{1} b^\top - Y_r\|_F^2 + \lambda \|W\|_F^2, \tag{14}$$

which has a standard closed-form solution after mean-centering. We fit both the forward map A_i^{out} and the reverse map A_i^{in}.

Practical anchor selection. Anchors should cover diverse semantics (reasoning, instruction-following, factual text, etc.) to avoid degenerate alignment on a narrow subspace. Because ridge regression is inexpensive, alignment can be re-fit whenever new models join the system.

C.3 Inference Protocols and Role Interaction

Unified "read–think–write" abstraction. All protocols can be expressed as repeated application of:

1. **Read memory:** decode aggregated $U_{\text{mem}}^{\text{ref}}$ into an injected image span for the current agent.
2. **Think:** run the frozen backbone conditioned on the injection to produce either (a) a latent rollout H_i (for intermediate roles), or (b) an output token sequence (for the final role).
3. **Write memory:** encode H_i into a new universal message U_i^{ref} and add it to memory.

Chained vs. independent-join collaboration. In *chained* collaboration, role $t{+}1$ reads all messages produced by roles $\leq t$, enabling iterative refinement. In *independent-join* collaboration, multiple roles run from the same initial context and the final agent reads a merged set of messages. Both modes preserve the same communication primitives and differ only in the memory update schedule.

Memory budgeting and bounded communication cost. Because each message consists of K_{u} universal tokens and decoding always writes into a fixed image span, the communication cost is bounded: message extraction costs $O(T)$ steps (fixed T), and message consumption costs $O(L_{\text{img}}^{(i)})$ (fixed by the receiver's VLM design). This contrasts with text communication

where message length—and thus communication and decoding overhead—can grow with content verbosity.

D Why Can a Tiny Codec and Few Anchors Work?

This section provides intuition for two empirical observations: (1) a lightweight codec can reliably transmit rich semantics through a VLM's vision interface, and (2) a simple affine map can align universal tokens across heterogeneous model families using only a small anchor set.

D.1 The VLM vision span is already a continuous prompt interface

A central design pattern in modern VLMs is *token-level conditioning* of a language model on a sequence of projected visual features. Architectures such as Flamingo and BLIP-2 construct a set of image-conditioned embeddings and feed them to (or alongside) a language model as a prefix/context. (Alayrac et al., 2022; Li et al., 2023b)[2] This means the language backbone is trained (or adapted) to interpret a *dense sequence of continuous vectors* as meaningful context.

From this viewpoint, the Vision Wormhole does not ask the VLM to do something unnatural. It uses the existing interface for continuous conditioning—the image-token span—but repurposes it for *model-to-model* communication rather than image understanding.

D.2 Why a small codec can be sufficient: we are not learning semantics from scratch

The codec is small, yet it works because it is *not* tasked with learning language or world knowledge. Those capabilities live in the frozen VLM backbone. Instead, the codec learns a *re-parameterization*: given an internal summary of a message (the latent rollout), produce a continuous prompt (the injected image span) that induces approximately the same downstream behavior as if the message had been provided in text.

There are three reasons this mapping can have relatively low complexity:

(i) Contrastive pretraining makes visual representations "text-like". Many VLM pipelines begin with contrastive image-text pretraining (e.g., CLIP, ALIGN, SigLIP), which explicitly aligns visual and textual semantics in a shared embedding geometry. (Radford et al., 2021; Jia et al., 2021; Zhai et al., 2023) As a result, the projected vision tokens that condition the language model often inhabit a semantic space compatible with language inference. This makes the image-token span a natural carrier for non-visual semantic content.

(ii) Continuous prompts can steer large frozen models with very few parameters. A line of work on soft prompting shows that learning a small set of continuous vectors is often sufficient to condition a frozen language model to perform new tasks (Prompt Tuning; Prefix-Tuning). (Lester et al., 2021; Li & Liang, 2021) Our injected vision-span perturbation plays an analogous role: it is a continuous prompt that steers a frozen backbone. The codec simply learns to *generate* such a prompt from a sender-side latent summary.

(iii) Distillation yields dense supervision per anchor. Even with a small number of anchors, self-distillation is information-rich: each anchor provides (a) a high-dimensional target hidden state and (b) a full next-token distribution over the vocabulary. This is far more informative than a single scalar label. Moreover, by extracting a latent rollout, each anchor induces a structured input sequence H_i rather than a single vector, providing additional learning signal without requiring longer text.

[2] We cite Flamingo and BLIP-2 as representative examples of the broader "visual tokens as prompt" paradigm.

Table 6: **Mid-sized heterogeneous MAS model configurations (4B–12B).** Four roles alternate across a four-step workflow (Planner → Critic → Refiner → Judger).

Backbone Setup	Backbones	Role assignment (Planner, Critic, Refiner, Judger)
2 backbones	Qwen3-VL-8B + Gemma-3-12B	(Qwen3-VL-8B, Gemma-3-12B, Qwen3-VL-8B, Gemma-3-12B)
2 backbones	Qwen3-VL-8B + Gemma-3-4B	(Gemma-3-4B, Qwen3-VL-8B, Gemma-3-4B, Qwen3-VL-8B)

D.3 Why few anchors can align models with an affine map

A working hypothesis: universal tokens factor into semantics + model-specific basis. Suppose there exists an underlying semantic representation $z(m) \in \mathbb{R}^D$ for message m that is approximately shared across models, while each model's encoder produces universal tokens in a model-specific coordinate system:

$$U_i(m) \;\approx\; \text{reshape}\big(z(m)R_i\big) + \epsilon, \tag{15}$$

where $R_i \in \mathbb{R}^{D\times D}$ is an (approximately) invertible linear transform and ϵ is residual noise. Under this model, mapping $U_i(m)$ into a reference space amounts to estimating R_i^{-1} (up to translation), which is exactly what ridge regression in Eq. (6) does.

Empirical precedent: linear alignment and model stitching. The idea that representations across networks can be related by simple learned mappings appears in multiple settings. In computer vision, early work studied equivalences and alignments between representation spaces. (Lenc & Vedaldi, 2015) More recently, model stitching investigates connecting intermediate representations of different networks with small modules and observes surprising transferability in practice. (Bansal et al., 2021) While these results do not prove linear equivalence in general, they provide precedent that nontrivial cross-model translation can sometimes be achieved with lightweight mappings when models share training signals and inductive biases.

Why universal-tokenization makes alignment easier than raw hidden states. Directly aligning raw hidden states across heterogeneous backbones is hard because those states mix many factors (tokenization, positional conventions, layerwise dynamics). Our encoder $\mathcal{E}_i$ is trained to compress a rollout into a *small, structured token set* under a distillation objective. This encourages U_i to represent information that the frozen backbone actually uses for prediction, while discarding idiosyncratic nuisance variation. In effect, $\mathcal{E}_i$ acts as a learned "bottleneck" that makes the remaining cross-model mismatch closer to an affine change-of-basis.

D.4 Limits and failure modes (what this analysis does not claim)

This reasoning supports why small codecs and few anchors can work in many practical cases, but it does not imply universal success. The affine hypothesis in Eq. (15) can break when model families differ substantially in training data, tokenization conventions, multimodal fusion design, or calibration of embedding norms. In such cases, more expressive alignment maps or additional anchors may be required.

E Mid-sized Heterogeneous MAS (4B–12B)

This section reports a preliminary larger-model two-backbone setting (4B–12B) that is separated from the main experiments in the main text. We intentionally keep the *same* codec configuration as the main setting—including latent rollout length (1024 steps) and image-side injection budget (256 visual tokens)—to test whether the default bandwidth transfers to stronger backbones without retuning.

Table 7: **Mid-sized heterogeneous MAS results across datasets.** We report accuracy (%), wall-clock time (s/query, batch-normalized), and improvement (ΔAcc in pp, speedup in ×) of VW vs Text.

	P/R: Gemma-3-4B, C/J: Qwen3-VL-8B						P/R: Qwen3-VL-8B, C/J: Gemma-3-12B					
Dataset	**Text Acc**	**Text Time**	**VW Acc**	**VW Time**	**ΔAcc**	**Speedup**	**Text Acc**	**Text Time**	**VW Acc**	**VW Time**	**ΔAcc**	**Speedup**
GSM8K	88.0%	54.2s	89.4%	20.3s	+1.4pp	2.67×	92.2%	82.6s	88.9%	12.7s	-3.3pp	6.50×
ARC-Easy	97.1%	60.1s	97.7%	19.2s	+0.6pp	3.13×	98.2%	75.9s	97.7%	10.5s	-0.5pp	7.23×
ARC-Challenge	93.1%	62.7s	93.0%	24.4s	-0.1pp	2.57×	95.6%	87.4s	93.5%	11.9s	-2.1pp	7.34×
GPQA	58.1%	477.9s	58.1%	196.0s	+0.0pp	2.44×	61.6%	872.5s	39.9%	53.2s	-21.7pp	16.40×
MedQA	80.0%	139.6s	78.3%	68.3s	-1.7pp	2.04×	78.0%	216.1s	73.0%	20.4s	-5.0pp	10.59×
MBPP-Plus	65.9%	200.0s	67.2%	73.0s	+1.3pp	2.74×	77.8%	237.8s	73.8%	17.6s	-4.0pp	13.51×
HumanEval-Plus	69.5%	207.8s	68.9%	89.1s	-0.6pp	2.33×	90.2%	253.2s	74.4%	25.4s	-15.8pp	9.97×
AIME 2024	66.7%	1522.2s	53.3%	534.0s	-13.4pp	2.85×	60.0%	1639.1s	26.7%	99.1s	-33.3pp	16.54×
AIME 2025	60.0%	1409.8s	40.0%	569.9s	-20.0pp	2.47×	46.7%	1813.3s	20.0%	130.1s	-26.7pp	13.94×

Table 8: **Single-agent baseline accuracy by backbone and dataset.** Values are accuracy (%) from the standalone single-agent runs in `summary.json`; Macro Avg. is the mean across the listed datasets for each model.

Model	GSM8K	ARC-Easy	ARC-Challenge	GPQA	MedQA	MBPP-Plus	HumanEval-Plus	AIME 2024	AIME 2025	**Macro Avg.**
Qwen3-VL-2B	74.8	90.9	80.6	34.9	43.7	51.1	37.8	23.3	20.0	50.8
Gemma-3-4B	83.4	92.2	83.5	35.4	49.7	71.2	65.8	3.3	16.7	55.7
SmolVLM2-2.2B	40.9	43.6	32.9	27.3	29.0	33.3	25.0	0.0	0.0	25.8
LFM2.5-VL-1.6B	63.0	82.6	68.9	26.3	41.0	43.6	36.6	0.0	0.0	40.2

E.1 Results and Analysis

In this preliminary mid-sized setting (Table 7), VW still provides strong efficiency gains: macro-average speedup is 5.92×, and several difficult datasets exceed 10× speedup. At the same time, we observe accuracy degradation on multiple complex tasks (especially GPQA and AIME), yielding a macro-average ΔAcc of −5.3pp.

A plausible explanation is that the default fixed bandwidth (1024 latent steps, 256 visual tokens) is sufficient for smaller-model settings but becomes a bottleneck for stronger backbones that produce richer intermediate reasoning states. This does *not* imply a hard 256-token limit for Vision Wormhole: the communication bandwidth can be increased by expanding the visual span, e.g., stacking multiple images or using larger image resolutions when supported by the VLM. For example, one can target an effective ∼1024-token visual span with multiple standard-resolution images (e.g., four 224 × 224 images in Gemma-style prompts) or a single higher-resolution image in models that support longer single-image tokenization (e.g., ∼1008-pixel side length in Qwen-style prompts).

F Detailed Single vs MAS

Section 4.4 reports model-level aggregates. We first report explicit single-agent baseline accuracies for each backbone, then provide the full dataset- and configuration-level breakdown.

Each baseline value in Table 8 comes from a dedicated single-agent run for that model (no multi-agent orchestration). This makes it easy to directly compare standalone capability against the combined-MAS results in the next tables.

Here we provide the full dataset- and configuration-level breakdown. Each table is centered on one baseline backbone and lists all heterogeneous MAS configurations that include that backbone. For each dataset, we report the single-agent baseline and, per MAS configuration, TextMAS (T) and Vision Wormhole (V) accuracy.

Detailed observations. The per-configuration view reinforces the main trend: TextMAS frequently drops below the single-agent baseline on stronger backbones, while VW remains closer to parity across many datasets and settings. This is consistent with the broader finding that heterogeneous MAS can be hurt by coordination and aggregation effects, especially

Table 9: **Detailed single-agent vs heterogeneous MAS for Qwen3-VL-2B.** Accuracy (%) by dataset and MAS configuration. MAS cells report TextMAS (T) and Vision Wormhole (V). Green/red indicates above/below the single-agent baseline for that dataset.

Dataset	Single	P/R: Gemma-3-4B C/J: Qwen3-VL-2B (Std)	P/R: Gemma-3-4B C/J: Qwen3-VL-2B (WeakSup)	P/R: LFM2.5-VL-1.6B C/J: Qwen3-VL-2B	P/R: SmolVLM2-2.2B C/J: Qwen3-VL-2B (Std)	P/R: SmolVLM2-2.2B C/J: Qwen3-VL-2B (WeakSup)	P: SmolVLM2-2.2B, C: LFM2.5-VL-1.6B R: Gemma-3-4B, J: Qwen3-VL-2B
GSM8K	74.8	T: 83.0, V: 76.2	T: 80.8, V: 77.6	T: 70.9, V: 76.6	T: 64.3, V: 74.8	T: 65.1, V: 77.0	T: 62.8, V: 75.7
ARC-Easy	90.9	T: 94.0, V: 92.4	T: 93.4, V: 90.3	T: 91.4, V: 91.7	T: 89.5, V: 92.3	T: 88.6, V: 91.8	T: 85.0, V: 92.0
ARC-Challenge	80.6	T: 86.0, V: 82.1	T: 87.2, V: 80.5	T: 81.7, V: 81.7	T: 78.2, V: 81.7	T: 78.7, V: 81.3	T: 74.3, V: 81.2
GPQA	34.9	T: 37.9, V: 39.9	T: 29.8, V: 34.9	T: 42.4, V: 34.9	T: 34.9, V: 37.9	T: 32.3, V: 33.3	T: 33.8, V: 36.9
MedQA	43.7	T: 53.3, V: 48.0	T: 53.3, V: 45.0	T: 51.3, V: 49.7	T: 44.7, V: 47.0	T: 47.3, V: 49.0	T: 46.3, V: 47.7
MBPP-Plus	51.1	T: 50.5, V: 51.3	T: 51.3, V: 46.6	T: 45.0, V: 51.1	T: 40.5, V: 47.9	T: 37.8, V: 49.2	T: 28.8, V: 47.9
HumanEval-Plus	37.8	T: 40.9, V: 37.2	T: 45.1, V: 40.2	T: 38.4, V: 37.8	T: 31.1, V: 40.9	T: 31.1, V: 42.7	T: 19.5, V: 37.8
AIME 2024	23.3	T: 23.3, V: 36.7	T: 23.3, V: 26.7	T: 30.0, V: 20.0	T: 13.3, V: 23.3	T: 13.3, V: 36.7	T: 13.3, V: 26.7
AIME 2025	20.0	T: 33.3, V: 26.7	T: 16.7, V: 23.3	T: 13.3, V: 20.0	T: 20.0, V: 23.3	T: 16.7, V: 23.3	T: 26.7, V: 20.0
Macro Avg.	50.8	T: 55.8, V: 54.5	T: 53.4, V: 51.7	T: 51.6, V: 51.5	T: 46.3, V: 52.1	T: 45.7, V: 53.8	T: 43.4, V: 51.8

Table 10: **Detailed single-agent vs heterogeneous MAS for Gemma-3-4B.** Accuracy (%) by dataset and MAS configuration. MAS cells report TextMAS (T) and Vision Wormhole (V). Green/red indicates above/below the single-agent baseline for that dataset.

Dataset	Single	P/R: Gemma-3-4B C/J: Qwen3-VL-2B (Std)	P/R: Gemma-3-4B C/J: Qwen3-VL-2B (WeakSup)	P/R: Gemma-3-4B C/J: Qwen3-VL-8B	P/R: LFM2.5-VL-1.6B C/J: Gemma-3-4B	P/R: SmolVLM2-2.2B C/J: Gemma-3-4B	P: SmolVLM2-2.2B, C: LFM2.5-VL-1.6B R: Gemma-3-4B, J: Qwen3-VL-2B
GSM8K	83.4	T: 83.0, V: 76.2	T: 80.8, V: 77.6	T: 88.0, V: 89.4	T: 71.7, V: 85.1	T: 67.8, V: 85.4	T: 62.8, V: 75.7
ARC-Easy	92.2	T: 94.0, V: 92.4	T: 93.4, V: 90.3	T: 97.1, V: 97.7	T: 88.6, V: 90.8	T: 84.4, V: 90.2	T: 85.0, V: 92.0
ARC-Challenge	83.5	T: 86.0, V: 82.1	T: 87.2, V: 80.5	T: 93.1, V: 93.0	T: 77.0, V: 81.1	T: 70.6, V: 80.7	T: 74.3, V: 81.2
GPQA	35.4	T: 37.9, V: 39.9	T: 29.8, V: 34.9	T: 58.1, V: 58.1	T: 31.3, V: 24.2	T: 26.3, V: 29.3	T: 33.8, V: 36.9
MedQA	49.7	T: 53.3, V: 48.0	T: 53.3, V: 45.0	T: 80.0, V: 78.3	T: 47.7, V: 52.3	T: 41.0, V: 48.3	T: 46.3, V: 47.7
MBPP-Plus	71.2	T: 50.5, V: 51.3	T: 51.3, V: 46.6	T: 65.9, V: 67.2	T: 45.8, V: 66.4	T: 44.7, V: 67.7	T: 28.8, V: 47.9
HumanEval-Plus	65.8	T: 40.9, V: 37.2	T: 45.1, V: 40.2	T: 69.5, V: 68.9	T: 43.9, V: 60.4	T: 32.9, V: 59.1	T: 19.5, V: 37.8
AIME 2024	3.3	T: 23.3, V: 36.7	T: 23.3, V: 26.7	T: 66.7, V: 53.3	T: 0.0, V: 6.7	T: 3.3, V: 6.7	T: 13.3, V: 26.7
AIME 2025	16.7	T: 33.3, V: 26.7	T: 16.7, V: 23.3	T: 60.0, V: 40.0	T: 3.3, V: 13.3	T: 3.3, V: 10.0	T: 26.7, V: 20.0
Macro Avg.	55.7	T: 55.8, V: 54.5	T: 53.4, V: 51.7	T: 75.4, V: 71.8	T: 45.5, V: 53.4	T: 41.6, V: 53.0	T: 43.4, V: 51.8

in role-sensitive stages such as judging (Pappu et al., 2026). Even when weaker models are part of the same pipeline, VW remains comparatively stable and preserves more of the strong model's baseline capability than text-only exchange in many cases.

G Time Distribution Analysis

We report a complementary time-distribution analysis using per-example end-to-end wall-clock time measurements recorded for each run. In our logging, per-example time is computed as `batch_time / batch_samples`, so these plots characterize batch-normalized runtime (seconds/query) rather than interactive tail latency. For every dataset and model configuration, we compare TextMAS against Vision Wormhole (VW) under matched prompts, roles, and decoding budgets. Panels visualize (i) runtime distributions and (ii) pooled boxplots. Blue denotes TextMAS and orange denotes VW. Within each panel page, rows correspond to datasets and columns correspond to model configurations.

Key observations. Across regimes, VW generally shifts the runtime distributions left (faster inference) and makes them more concentrated. In TextMAS, communication is realized through variable-length natural-language messages; message verbosity and the receiver's re-encoding overhead introduce substantial run-to-run and example-to-example variability, often manifesting as heavier right tails in the histograms. By contrast, VW transmits information through a fixed-size vision-token span with a fixed latent-step budget, which bounds the communication bandwidth and reduces variability induced by message length. This variance reduction is visible in the pooled boxplots as smaller interquartile ranges and fewer extreme slow cases, especially on long-budget tasks where text communication can amplify decoding overhead.

Reading the panels. To accommodate the wide dynamic range of end-to-end runtimes across tasks, distribution panels use log-scaled time axes (in seconds). These distribution panels highlight multimodality and long-tail effects. Boxplots pool per-example times within each dataset–configuration cell, then apply a *pooled* z-score normalization using the

Table 11: **Detailed single-agent vs heterogeneous MAS for SmolVLM2-2.2B.** Accuracy (%) by dataset and MAS configuration. MAS cells report TextMAS (T) and Vision Wormhole (V). Green/red indicates above/below the single-agent baseline for that dataset.

Dataset	Single	P/R: SmolVLM2-2.2B C/J: Gemma-3-4B	P/R: SmolVLM2-2.2B C/J: Qwen3-VL-2B (Std)	P/R: SmolVLM2-2.2B C/J: Qwen3-VL-2B (WeakSup)	P: SmolVLM2-2.2B, C: LFM2.5-VL-1.6B R: Gemma-3-4B, J: Qwen3-VL-2B
GSM8K	40.9	T: 67.8, V: 85.4	T: 64.3, V: 74.8	T: 65.1, V: 77.0	T: 62.8, V: 75.7
ARC-Easy	43.6	T: 84.4, V: 90.2	T: 89.5, V: 92.3	T: 88.6, V: 91.8	T: 85.0, V: 92.0
ARC-Challenge	32.9	T: 70.6, V: 80.7	T: 78.2, V: 81.7	T: 78.7, V: 81.3	T: 74.3, V: 81.2
GPQA	27.3	T: 26.3, V: 29.3	T: 34.9, V: 37.9	T: 32.3, V: 33.3	T: 33.8, V: 36.9
MedQA	29.0	T: 41.0, V: 48.3	T: 44.7, V: 47.0	T: 47.3, V: 49.0	T: 46.3, V: 47.7
MBPP-Plus	33.3	T: 44.7, V: 67.7	T: 40.5, V: 47.9	T: 37.8, V: 49.2	T: 28.8, V: 47.9
HumanEval-Plus	25.0	T: 32.9, V: 59.1	T: 31.1, V: 40.9	T: 31.1, V: 42.7	T: 19.5, V: 37.8
AIME 2024	0.0	T: 3.3, V: 6.7	T: 13.3, V: 23.3	T: 13.3, V: 36.7	T: 13.3, V: 26.7
AIME 2025	0.0	T: 3.3, V: 10.0	T: 20.0, V: 23.3	T: 16.7, V: 23.3	T: 26.7, V: 20.0
Macro Avg.	25.8	T: 41.6, V: 53.0	T: 46.3, V: 52.1	T: 45.7, V: 53.8	T: 43.4, V: 51.8

Table 12: **Detailed single-agent vs heterogeneous MAS for LFM2.5-VL-1.6B.** Accuracy (%) by dataset and MAS configuration. MAS cells report TextMAS (T) and Vision Wormhole (V). Green/red indicates above/below the single-agent baseline for that dataset.

Dataset	Single	P/R: LFM2.5-VL-1.6B C/J: Gemma-3-4B	P/R: LFM2.5-VL-1.6B C/J: Qwen3-VL-2B	P: SmolVLM2-2.2B, C: LFM2.5-VL-1.6B R: Gemma-3-4B, J: Qwen3-VL-2B
GSM8K	63.0	T: 71.7, V: 85.1	T: 70.9, V: 76.6	T: 62.8, V: 75.7
ARC-Easy	82.6	T: 88.6, V: 90.8	T: 91.4, V: 91.7	T: 85.0, V: 92.0
ARC-Challenge	68.9	T: 77.0, V: 81.1	T: 81.7, V: 81.7	T: 74.3, V: 81.2
GPQA	26.3	T: 31.3, V: 24.2	T: 42.4, V: 34.9	T: 33.8, V: 36.9
MedQA	41.0	T: 47.7, V: 52.3	T: 51.3, V: 49.7	T: 46.3, V: 47.7
MBPP-Plus	43.6	T: 45.8, V: 66.4	T: 45.0, V: 51.1	T: 28.8, V: 47.9
HumanEval-Plus	36.6	T: 43.9, V: 60.4	T: 38.4, V: 37.8	T: 19.5, V: 37.8
AIME 2024	0.0	T: 0.0, V: 6.7	T: 30.0, V: 20.0	T: 13.3, V: 26.7
AIME 2025	0.0	T: 3.3, V: 13.3	T: 13.3, V: 20.0	T: 26.7, V: 20.0
Macro Avg.	40.2	T: 45.5, V: 53.4	T: 51.6, V: 51.5	T: 43.4, V: 51.8

mean and standard deviation computed over the *combined* (TextMAS+VW) time set for that cell. This shared normalization preserves relative shifts between methods while making cross-task variance comparisons more interpretable.

G.1 Main Setting (including 4-agent pool)

These panels cover the main setting, including both two-agent pairs and the four-agent pool. Across most dataset–configuration cells, VW exhibits a tighter runtime spread, indicating both faster and more stable batch-normalized runtime under bounded communication.

G.2 Mid-sized (4B–12B)

These panels correspond to the preliminary larger-model setting with the same default codec bandwidth. VW still shows substantial runtime reduction, while the spread difference between methods varies by task, consistent with a stronger bandwidth-performance tradeoff in this regime.

G.3 Weakly supervised codec variant

These panels evaluate the weakly supervised codec variant trained with fewer anchors. The runtime profiles remain broadly similar to the main setting, with VW typically faster and less dispersed, supporting robustness of the communication mechanism under reduced supervision.

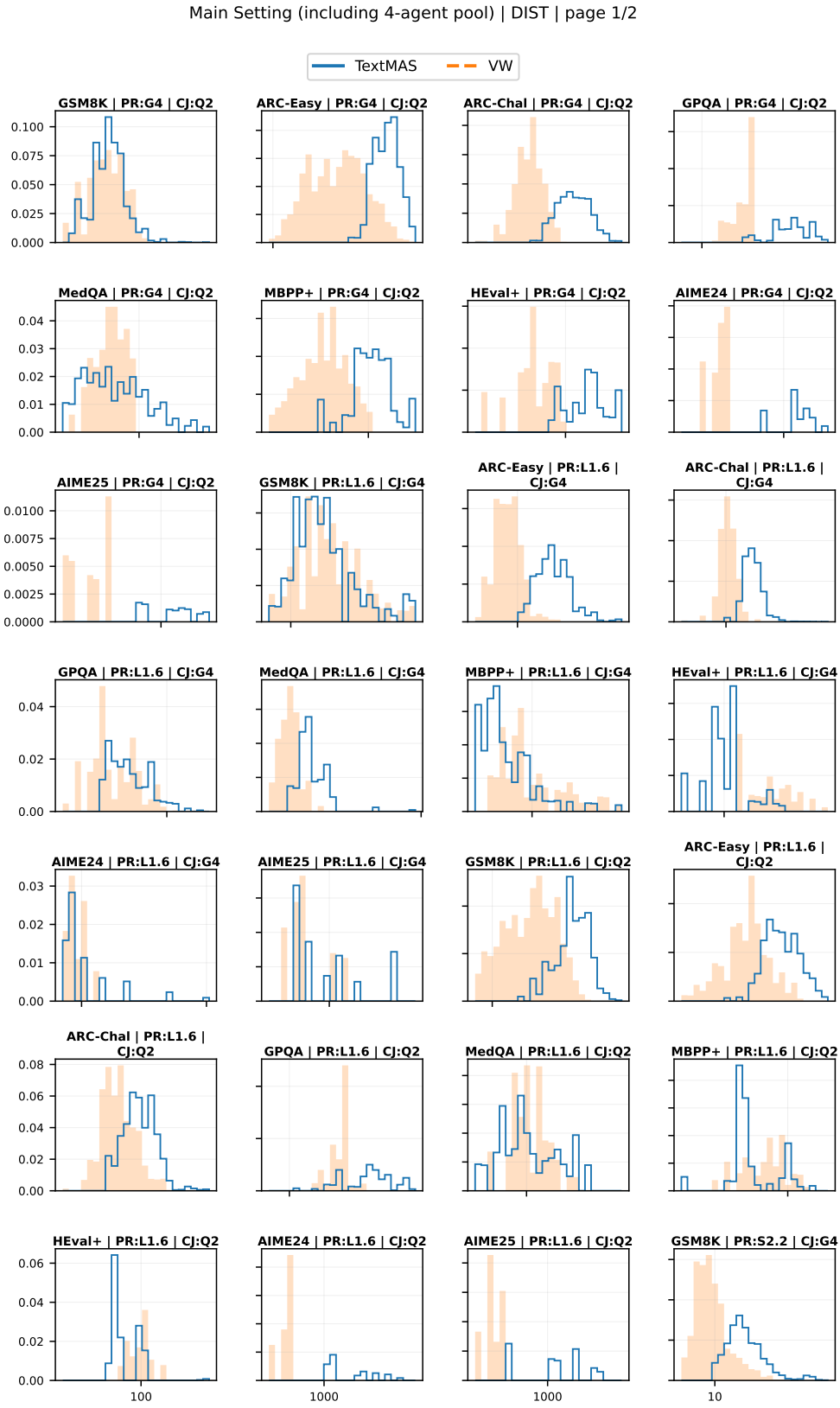

Figure 3: Distribution panels (histograms; log-scaled x-axis) of end-to-end wall-clock time (seconds/query, batch-normalized) for TextMAS vs VW (Blue=TextMAS, Orange=VW).

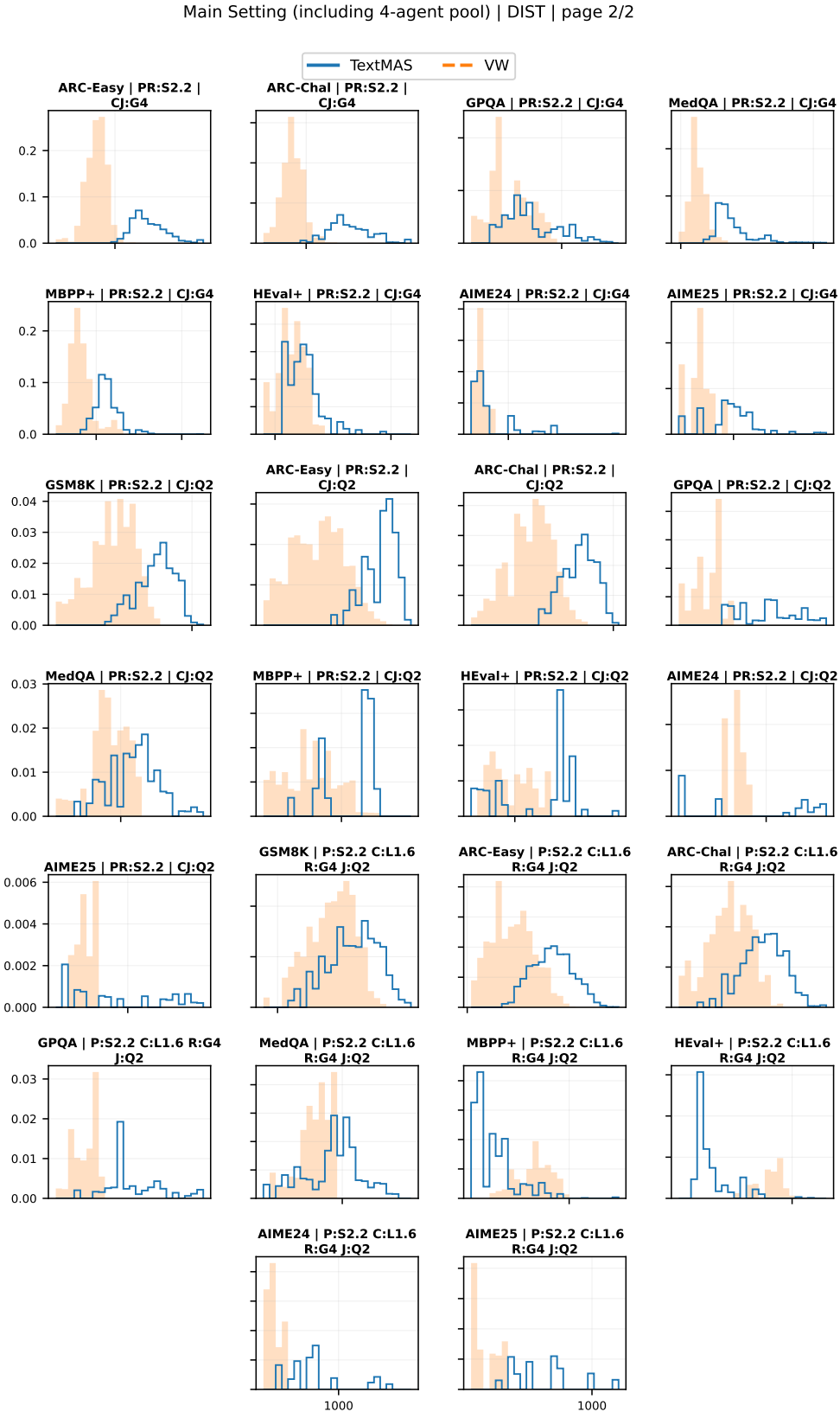

Figure 4: Distribution panels (histograms; log-scaled x-axis) of end-to-end wall-clock time (seconds/query, batch-normalized) for TextMAS vs VW (Blue=TextMAS, Orange=VW).

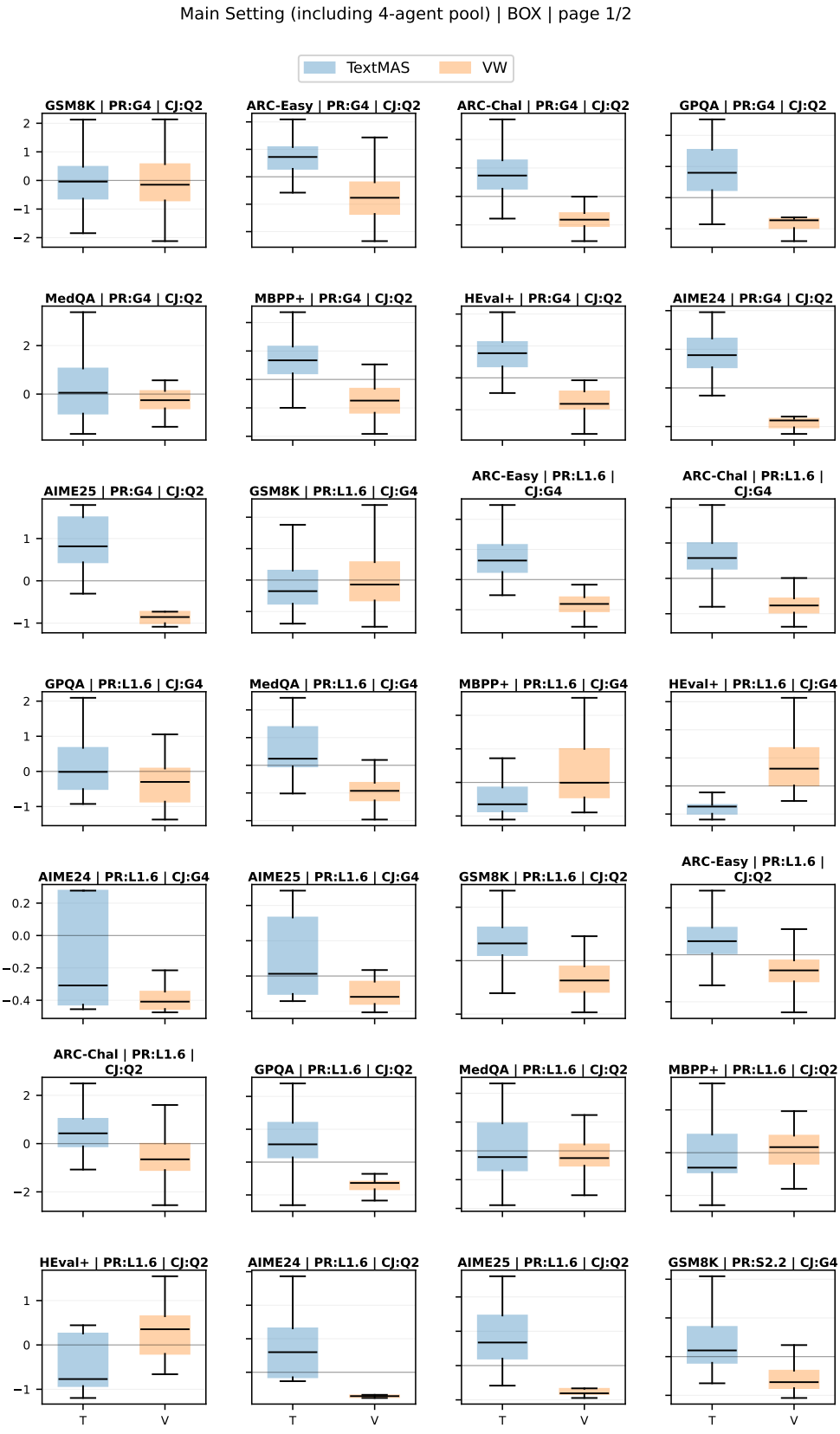

Figure 5: Pooled boxplot panels of end-to-end wall-clock time after pooled per-cell z-scoring (seconds/query, batch-normalized; Blue=TextMAS, Orange=VW).

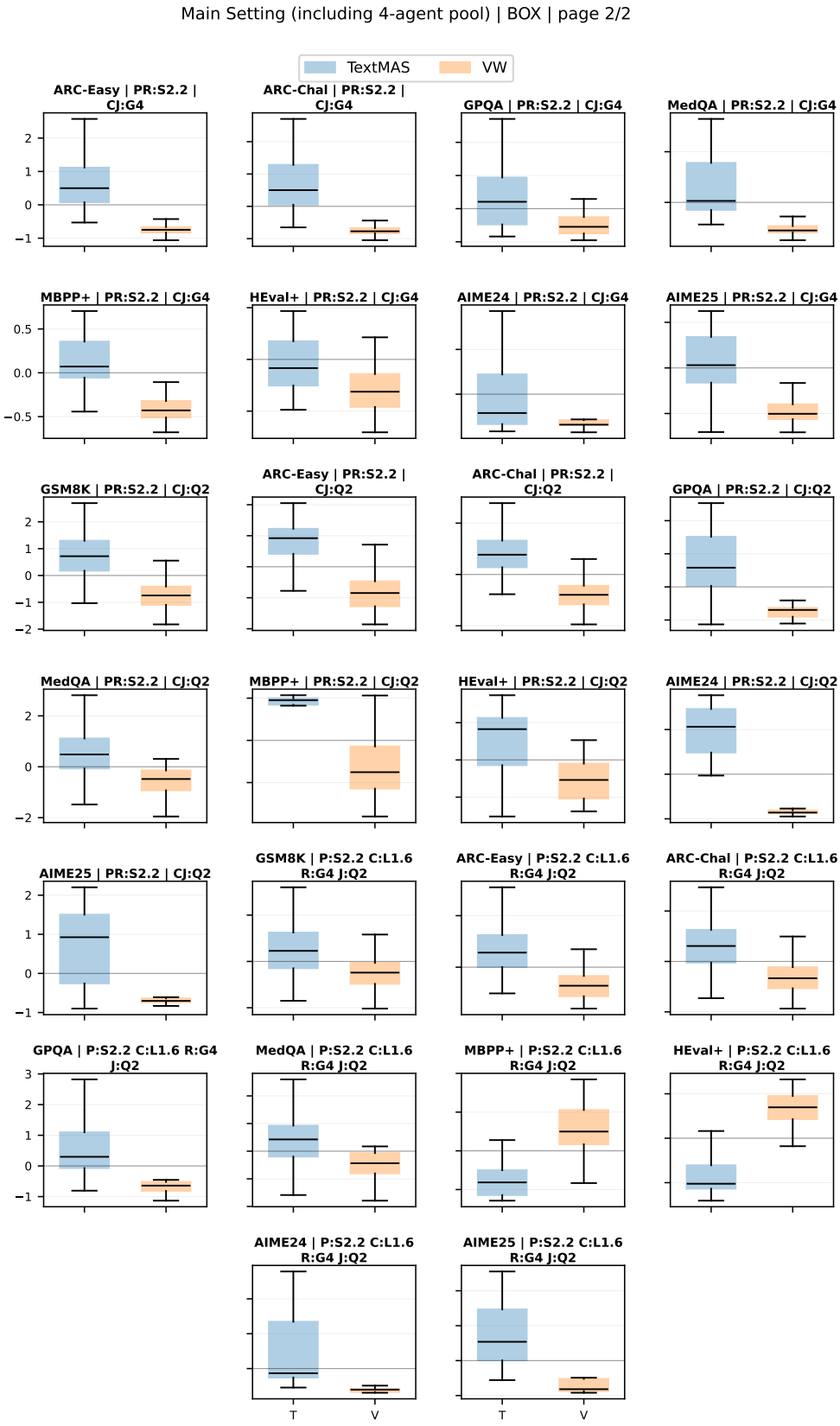

Figure 6: Pooled boxplot panels of end-to-end wall-clock time after pooled per-cell z-scoring (seconds/query, batch-normalized; Blue=TextMAS, Orange=VW).

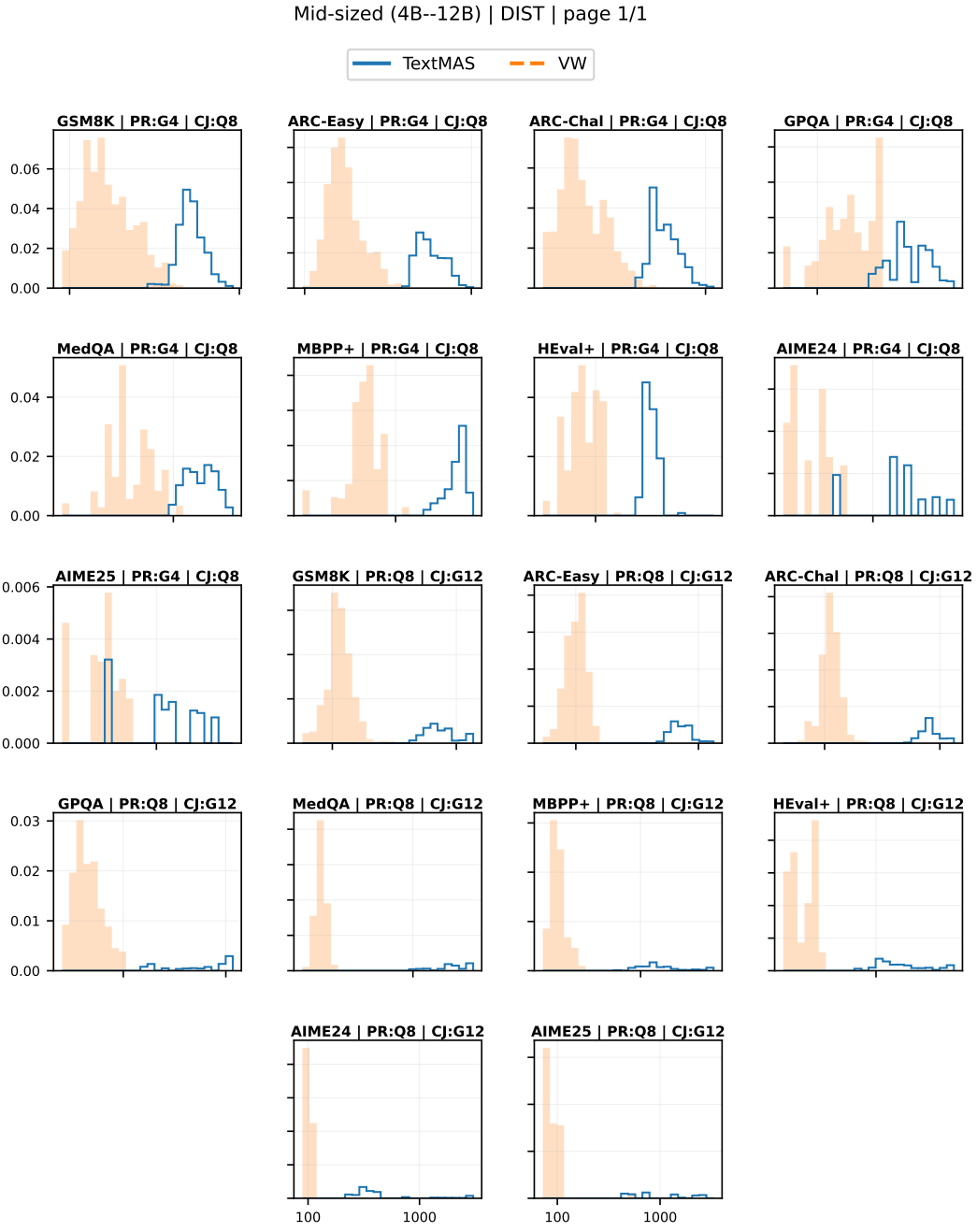

Figure 7: Distribution panels (histograms; log-scaled x-axis) of end-to-end wall-clock time (seconds/query, batch-normalized) for TextMAS vs VW (Blue=TextMAS, Orange=VW).

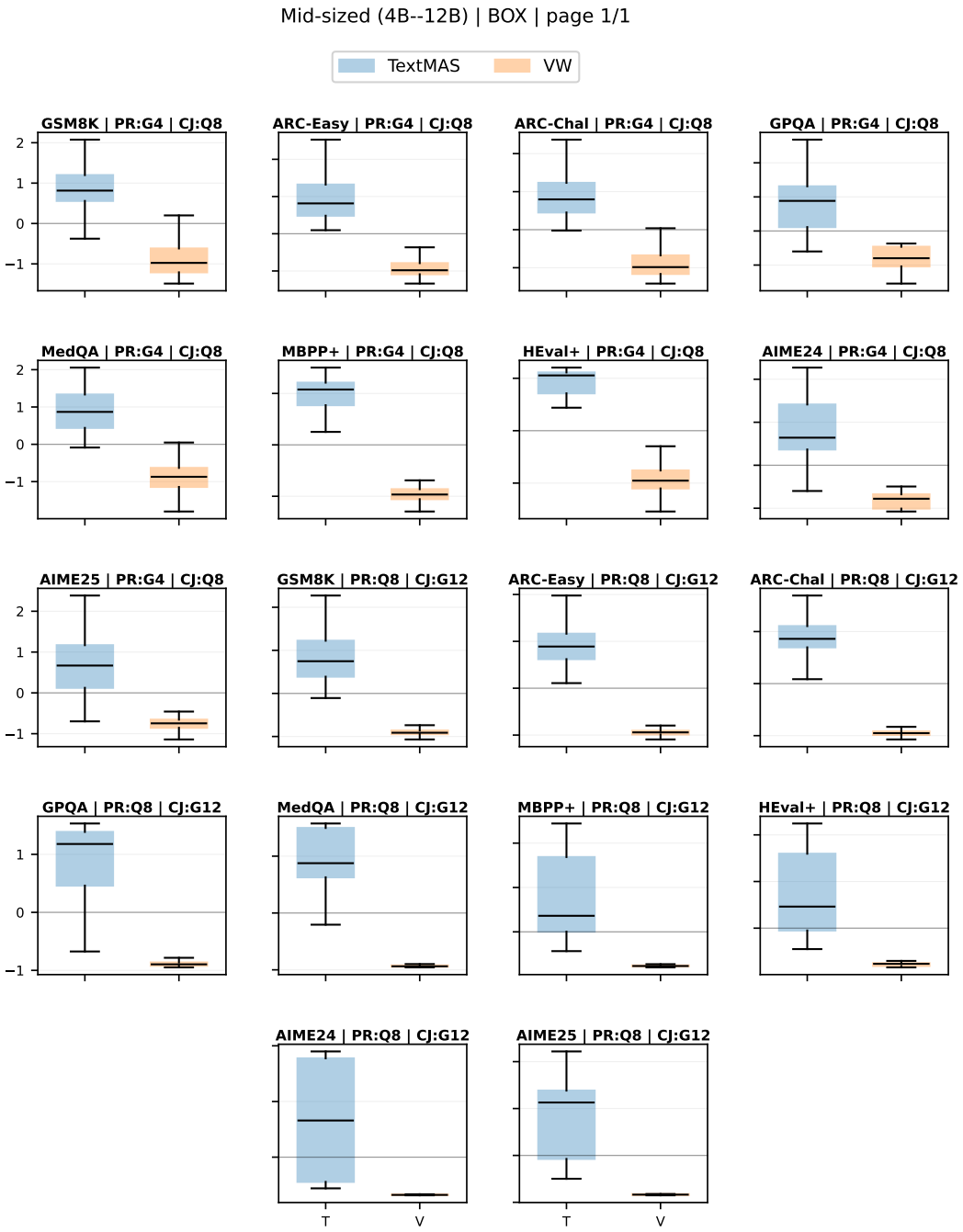

Figure 8: Pooled boxplot panels of end-to-end wall-clock time after pooled per-cell z-scoring (seconds/query, batch-normalized; Blue=TextMAS, Orange=VW).

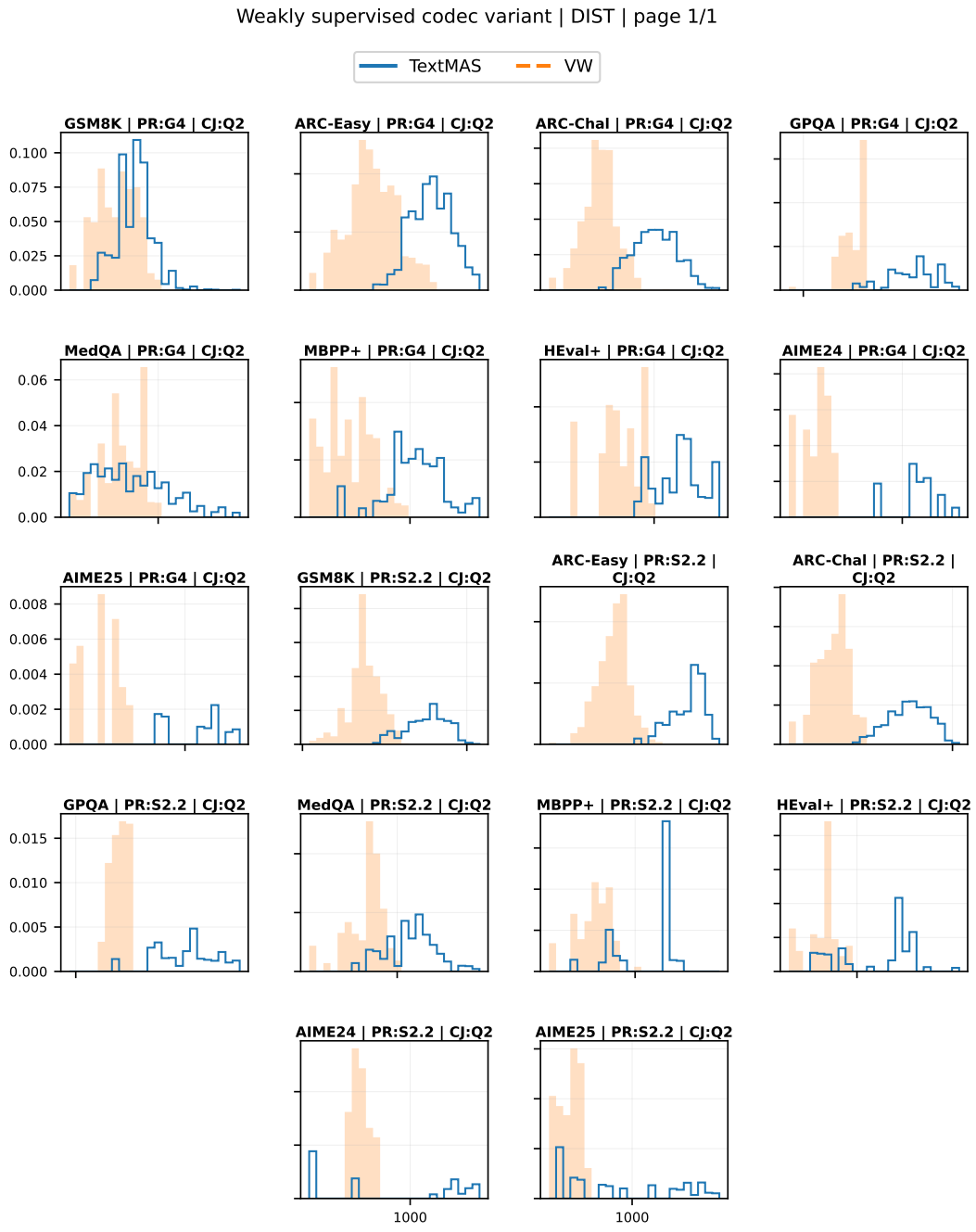

Figure 9: Distribution panels (histograms; log-scaled x-axis) of end-to-end wall-clock time (seconds/query, batch-normalized) for TextMAS vs VW (Blue=TextMAS, Orange=VW).

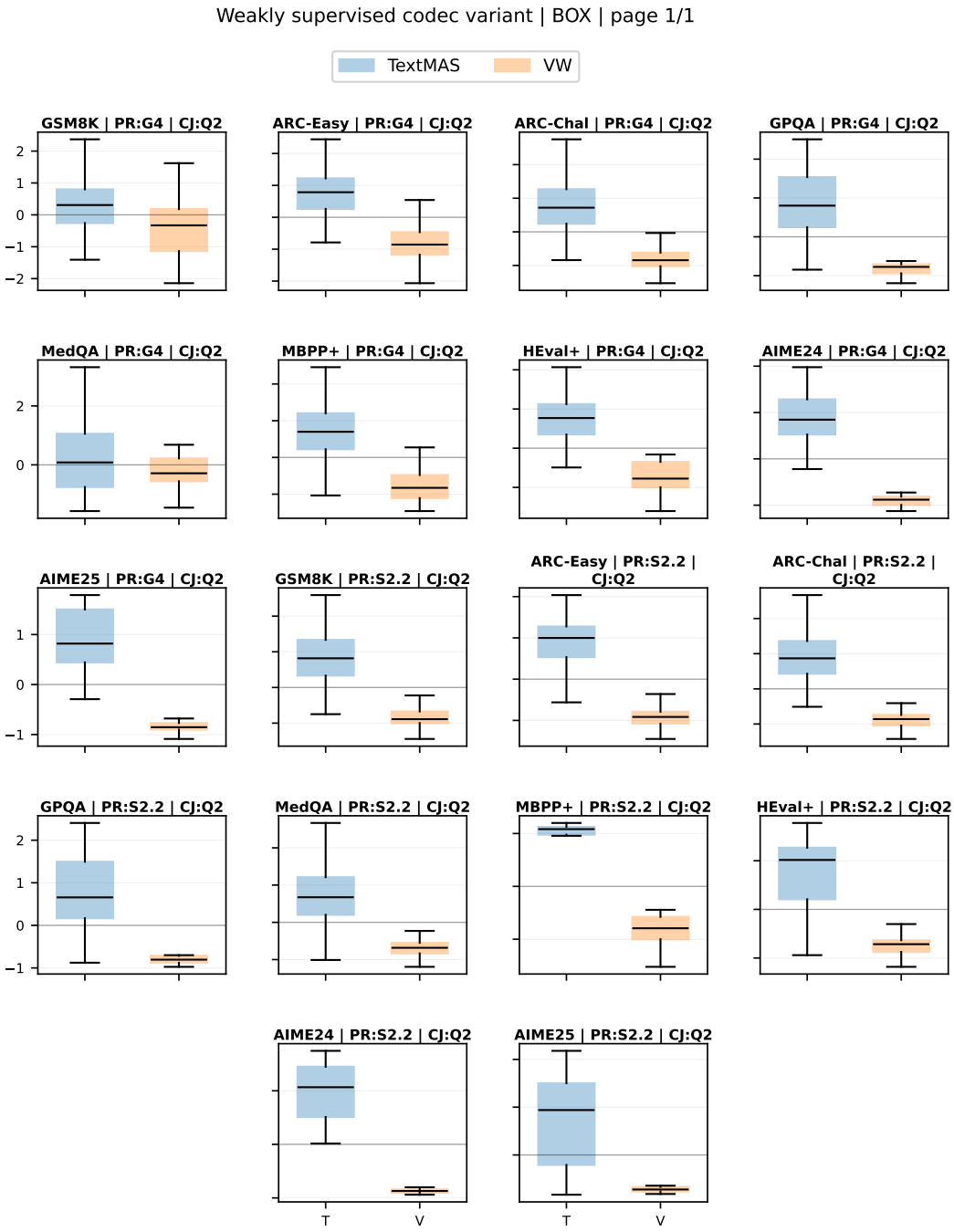

Figure 10: Pooled boxplot panels of end-to-end wall-clock time after pooled per-cell z-scoring (seconds/query, batch-normalized; Blue=TextMAS, Orange=VW).

H Prompts Used in Our Experiments

We follow the same prompt templates as LatentMAS (Zou et al., 2025), and reproduce them here for completeness. This appendix documents the exact prompt templates used to instantiate our sequential, role-based MAS protocol (Planner → Critic → Refiner → Judger). We report templates for (i) **TextMAS**, where inter-agent communication is carried by text context, and (ii) **Vision Wormhole (VW)**, where inter-agent messages are carried by latent messages injected through the vision-token span. In all prompts below, `<QUESTION>` denotes the task input and `<CONTEXT>` denotes the text-form message history from previous agents (when applicable).

System message. For non-Qwen backbones we use the default system message `You are a helpful assistant.`. For Qwen-family models we use `You are Qwen, created by Alibaba Cloud. You are a helpful assistant.`.

H.1 TextMAS (sequential, text-mediated)

```
You are a Planner Agent. Given an input question, design a clear, step-by
    -step plan for how to solve the question.

## Input Question:
<QUESTION>

Your outlined plan should be concise with a few bullet points for each
    step. Do not produce the final answer.

## Format your response as follows:
Planner Agent's Output:
[Your detailed plan here]

Now output your plan to solve the question below:
```

```
You are a Critic Agent. You are provided with:
(1) the original question, and
(2) the Planner Agent's plan in text format.

Your job is to carefully evaluate the correctness and completeness of the
     plan and provide helpful feedback.

## Input Question:
<QUESTION>

## Plan from Planner Agent:
<CONTEXT>

## Format your response as follows:
Critic Agent's Output:
Original Plan: [Copy the provided Planner Agent's plan here]
Feedback: [Your detailed feedback to improve the plan here]

Now, output your response below:
```

```
You are a Refiner Agent. You are provided with:
(1) the original question, and
(2) the Planner Agent's plan together with Critic Agent's feedback in
    text format.
```

```
Your job is to incorporate the feedback and produce an improved, refined
    step-by-step plan.

## Input Question:
<QUESTION>

## Original Plan and Critic Feedback:
<CONTEXT>

## Format your response as follows:
Refiner Agent's Output:
[Your refined and improved plan here]

Make sure your output plan is logically correct, concise, and sufficient
    to guide final problem solving.
Now, output your refined plan below:
```

```
Target Question: <QUESTION>

You are the final solver agent in a sequential multi-agent system (
    planner -> critic -> refiner -> solver).
You are provided with the Refiner Agent's plan as reference.

Refined Plan from Previous Agents:
<CONTEXT>

The plan might contain irrelevant or incorrect contents. Ignore them if
    they are not helpful for solving the target question.

You must reason step-by-step to solve the **provided Target Question**
    without outputting other irrelevant information.

At the end, output the final answer on a single line as: #### <number>.
```

```
Target Question: <QUESTION>

You are the final solver agent in a sequential multi-agent system (
    planner -> critic -> refiner -> solver).
You are provided with the Refiner Agent's plan as reference.

Refined Plan from Previous Agents:
<CONTEXT>

The plan might contain irrelevant or incorrect contents. Ignore them if
    they are not helpful for solving the target question.

You must reason step-by-step to solve the **provided Target Question**
    without outputting other irrelevant information.

Now, reason step by step and output the final answer inside \boxed{
    YOUR_FINAL_ANSWER}.
```

```
Target Question: <QUESTION>

You are the final solver agent in a sequential multi-agent system (
    planner -> critic -> refiner -> solver).
You are provided with the Refiner Agent's plan as reference.
```

```
Refined Plan from Previous Agents:
<CONTEXT>

The plan might contain irrelevant or incorrect contents. Ignore them if
    they are not helpful for solving the target question.

You must reason step-by-step to solve the **provided Target Question**
    without outputting other irrelevant information.
Your final answer must be selected from A,B,C,D. For example \boxed{A}.
    Do not add any other contents inside the box.

Now, reason step by step and output the final answer inside \boxed{
    YOUR_FINAL_ANSWER}.



Target Question: <QUESTION>

You are the final solver agent in a sequential multi-agent system (
    planner -> critic -> refiner -> solver).
You are provided with the Refiner Agent's plan as reference.

Refined Plan from Previous Agents:
<CONTEXT>

The plan might contain irrelevant or incorrect contents. Ignore them if
    they are not helpful for solving the target question.

You must reason step-by-step to solve the **provided Target Question**
    without outputting other irrelevant information.
You must put all python code as self-contained Python function(s) in
    markdown code blocks. For example:
```python
import math
def add(a, b):
    return a + b
```
Do not add any other contents inside the markdown code block.
```

H.2 Vision Wormhole (sequential, latent-mediated)

```
You are a Planner Agent. Given an input question, design a clear, step-by
    -step plan for how to solve the question.

Question: <QUESTION>

Your outlined plan should be concise with a few bulletpoints for each
    step. Do not produce the final answer.
Now output your plan to solve the question below:



Question: <QUESTION>

You are a Critic Agent to evaluate the correctness of the input plan for
    the given question and provide helpful feedback for improving the
    plan.
The plan information is provided in latent KV representation format.
    Review the plan and question and output:
```

```
(1) original plan contents
(2) constructive feedback on the original plan.

Format your response as follows:
Original Plan: [Copy the provided Planner Agent's plan here]
Feedback: [Your detailed feedback to improve the plan here]

Now, output your response below:



Question: <QUESTION>

You are a Refiner Agent to provide a refined step-by-step plan for
    solving the given question.
You are provided with:
(1) latent-format information: a previous plan with feedback
(2) text-format information: the input question you need to solve.

Based on the input, write a refined and improved plan to solve the
    question. Make sure your output plan is correct and concise.

Now, output your refined plan below:
```

Judger prompts. The VW Judger prompt is task-dependent and matches the TextMAS constraints (answer formatting, boxing, and code block requirements), with the only difference being that the judger is "provided with latent information for reference" rather than a text plan. We use the following task-specific Judger templates:

```
Target Question: <QUESTION>

You are a helpful assistant. You are provided with latent information for
     reference and a target question to solve.

The latent information might contain irrelevant contents. Ignore it if it
     is not helpful for solving the target question.

You must reason step-by-step to solve the provided Target Question
    without outputting other irrelevant information.

At the end, output the final answer on a single line as: #### <number>.



Target Question: <QUESTION>

You are a helpful assistant. You are provided with latent information for
     reference and a target question to solve.

The latent information might contain irrelevant contents. Ignore it if it
     is not helpful for solving the target question.

You must reason step-by-step to solve the provided Target Question
    without outputting other irrelevant information.

Now, reason step by step and output the final answer inside \boxed{
    YOUR_FINAL_ANSWER}.
```

```
Target Question: <QUESTION>

You are a helpful assistant. You are provided with latent information for
     reference and a target question to solve.

The latent information might contain irrelevant contents. Ignore it if it
     is not helpful for solving the target question.

You must reason step-by-step to solve the provided Target Question
    without outputting other irrelevant information.
Your final answer must be selected from A,B,C,D. For example \boxed{A}.
    Do not add any other contents inside the box.

Now, reason step by step and output the final answer inside \boxed{
    YOUR_FINAL_ANSWER}.



Target Question: <QUESTION>

You are a helpful assistant. You are provided with latent information for
     reference and a target question to solve.

The latent information might contain irrelevant contents. Ignore it if it
     is not helpful for solving the target question.

You must reason step-by-step to solve the provided Target Question
    without outputting other irrelevant information.
You must put all python code as self-contained Python function in
    markdown code blocks. For example ```python
import math
def add(a, b):
    return a + b```. Do not add any other contents inside the markdown
        code block.

Now, reason step by step and output the final answer inside ```python
YOUR_PYTHON_CODE
```.
```
```