ESCAPING OVERSQUASHING: ADDRESSABLE AND SUPPORT-AWARE GLOBAL MEMORY FOR MESSAGE PASSING NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Virtual nodes are a natural tool against oversquashing: they replace long message-passing paths by a two-hop global route. But when many nodes share one global state, that shortcut can become a bottleneck itself. We study two properties of this global memory. First, *addressability*: under constant-margin address codes and a nonlinearity that amplifies this margin, multiplicative write/read maps provide M selectable memory rows with only $O(\log M)$ address-code dimensions. Cross-attention slots and a constrained $\mathrm{ELU}+1$ bilinear memory both satisfy these conditions. Second, *support awareness*: normalized cross-attention has no self-key for a latent query to use as a reference. A learned private anchor supplies this reference, keeps the read bounded, and exposes the strength of the matching source mass. We demonstrate the merits of such properties on several instances of Two-Radius and Tree-NeighborsMatch: both addressable realizations solve the controlled tasks through depth 5, where pooled VNs of comparable or larger size reach about 10.6%.

1 INTRODUCTION

Message-passing neural networks (MPNNs) keep computation local and sparse (Gilmer et al., 2017). The price is that a node only sees far-away information after it has travelled through many intermediate states. When many signals must share the same few states, they get compressed and partly lost. This is *oversquashing* (Alon & Yahav, 2021; Topping et al., 2022; Di Giovanni et al., 2023).

A virtual node (VN) is a simple way to shorten these paths: one extra state is connected to every node, so any two nodes can communicate through a two-hop global route (Gilmer et al., 2017; Ishiguro et al., 2019). Shortening the path does not by itself determine how much information the route can carry. If many dependencies share one global state, the state and its readout can become a new communication bottleneck. The relevant question is therefore not only whether the shortcut is short, but whether its memory can be written and queried efficiently.

This motivates using controlled oversquashing benchmarks to study the memory interface of the shortcut. Two-Radius (Mishayev et al., 2025) fixes the source–target distance while increasing the number of associations that must cross the cut, isolating finite-state capacity and retrieval. Tree-NeighborsMatch (Alon & Yahav, 2021) increases both path length and the number of candidate messages. Adding a global route removes the long-path component; the remaining question is how effectively the auxiliary state preserves and retrieves the required associations.

We ask what makes a global shortcut an *addressable memory*, rather than only a pooled global state. Different sources should be able to write distinct contents, and each target should be able to select the content it needs. We show that this addressable interface needs only three generic ingredients, so cross-attention is one realization rather than the definition itself. We also measure what an ordinary additive VN pays to emulate the same addressing operation. The second property is *support awareness*: a normalized read keeps relative weights but can lose how strongly the memory matches a query. We add a private reference inside the normalization to keep this information.

Contributions.

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

- **A general account of addressable memory.** Margin codes in $p = O(\log M)$ dimensions, a margin-amplifying nonlinearity, and multiplicative write/read maps give M selectable memory rows with explicit leakage bounds. Exact one-hot responses from an affine address map require $p \geq M - 1$ (Section 3).
- **A scoped cost result for implicit addressing.** Within the dense additive architecture formalized in Proposition 3.8, exact recovery of arbitrary continuous payloads requires write and read bottlenecks of width at least Md, hence $(Md)^2$ weights in the corresponding dense layers. Multiplicative memories make the same routing product explicit with shared projections whose parameter count is near-linear in M for fixed node and payload widths.
- **Support-aware normalization.** A normalized read can hide the absolute support behind its relative weights. A private reference keeps the read bounded while exposing this support; cardinality recovery is a special case (Section 4).
- **Controlled and real-graph experiments.** Two-Radius, Tree-NeighborsMatch, and semantic routing separate addressability from raw memory size and optimization effects. Standard LRGB tasks provide a transfer check in settings where the global bottleneck is not expected to dominate (Section 5).

Dense attention between all nodes would also give a global path, but at cost $O(n^2 d)$. We keep the MPNN as the local processor and add a bounded memory with $O(nMd)$ cost for M rows.

2 Virtual Nodes and the Oversquashing Cut

Two-Radius. Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be sources and $\mathcal{T} = \{t_1, \ldots, t_n\}$ targets. A few central nodes are adjacent to all of them, with no direct source–target edge. Source s_i carries $(i, \pi(i))$ for a permutation π of $[n]$, and target t_j must output $y_{t_j} = \pi(j)$. Every target is two hops from every source, so a VN adds no shorter path. Yet all n associations must cross the cut between sources and targets, and the targets must tell $n!$ assignments apart. A VN is a second route across the same cut (Figure 1); the question is how much it can carry and how well targets can read it.

We call the number of source-dependent states that can vary independently and be selected separately by the decoder the *effective factorization degree* L_{eff}. Identical VN clones count as one state (Appendix C); separately selectable memory rows count separately.

Theorem 2.1 (Finite-capacity requirement)**.** *Any deterministic architecture that solves every permutation-valued Two-Radius instance exactly, with effective factorization degree L_{eff}, width d, and b-bit coordinates, satisfies*

$$L_{\text{eff}} db \geq \log_2(n!). \tag{1}$$

Hence $d = \Omega(n \log n/(L_{\text{eff}} b))$.

The proof counts transcripts: different permutations need different source-dependent states (Appendix C). So raw size matters. But size is not enough: the targets must also be able to select their own entry.

Proposition 2.2 (A separable broadcast cannot implement arbitrary lookup)**.** *If the source-dependent contribution to target j has the form $\widehat{y}_j = f(h_j) + g(z)$, where h_j depends only on the target and z is one homogeneous global state, it cannot realize every Two-Radius instance.*

Swapping the labels of two targets p and q changes $g(z)$ in the same way for every target, while p and q need different changes. A joint decoder $F(h_j, z)$ can escape this, but only through an interaction between the target and the stored state. The rest of the paper makes this interaction explicit and measures its cost.

Tree-NeighborsMatch. Tree-NeighborsMatch (Alon & Yahav, 2021) places candidate labels at the leaves of a complete binary tree and gives each candidate a distinct local neighbor-count feature. The root receives one such count and must predict the label of the matching candidate. The count is supplied as an input feature in the benchmark, so the difficulty is communication rather than learning the count itself. Without a shortcut, information from 2^r candidates travels through the tree to the root. A global route shortens this communication, while still requiring the auxiliary state to preserve the candidate–label associations.

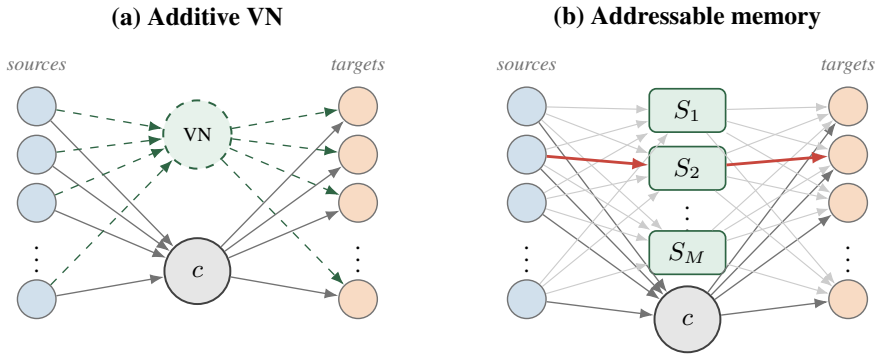

Figure 1: Two routes across the Two-Radius cut. **(a)** An additive VN adds one pooled state broadcast to every target. **(b)** An addressable multiplicative memory adds M separately selectable rows; the highlighted route is a source-specific write followed by a target-specific read. In both cases the graph path through the central node c is unchanged.

What a good shortcut needs. Both benchmarks remove the distance argument and keep the capacity one. They therefore test the two things a VN must do beyond being short: store many entries without mixing them, and let each target retrieve its own entry. The first is a matter of raw size (Theorem 2.1); the second is a matter of organization, which we call addressability. When sources can be repeated, a third need appears: the read must keep how much evidence supports it, which we call support awareness.

3 Addressable Global Memory

3.1 Additive and multiplicative memories

Let h_i be node states after a local MPNN block. Sources are mapped to routing keys $k_i = \eta_{\mathrm{w}}(h_i) \in \mathbb{R}^p$ and payloads $v_i \in \mathbb{R}^d$; targets are mapped to routing queries $q_j = \eta_{\mathrm{r}}(h_j) \in \mathbb{R}^p$.

Definition 3.1 (Additive and multiplicative memories)**.** *An* additive *memory pools $z = \sum_i \psi(h_i)$ (or the mean) with a generic map ψ and reads $F(h_j, z)$. An* addressable multiplicative *memory with M rows uses nonnegative address maps $\phi^{\mathrm{w}}, \phi^{\mathrm{r}} : \mathbb{R}^p \to \mathbb{R}^M_{\geq 0}$ that share a row index, and writes*

$$\widetilde{S} = \sum_i \phi^{\mathrm{w}}(k_i)\, v_i^\top \in \mathbb{R}^{M\times d}, \qquad Z = \sum_i \phi^{\mathrm{w}}(k_i) \in \mathbb{R}^M_{\geq 0}. \tag{2}$$

It reads either row-normalized, $o_j = \sum_m \rho_m(q_j)\, \widetilde{S}_m / Z_m$ with $\rho(q) = \phi^{\mathrm{r}}(q)/\mathbf{1}^\top \phi^{\mathrm{r}}(q)$, or jointly normalized, $o_j = \phi^{\mathrm{r}}(q_j)^\top \widetilde{S} \,/\, \phi^{\mathrm{r}}(q_j)^\top Z$.

In the idealized one-hot case, a multiplicative write is

$$\phi^{\mathrm{w}}(k_i)\, v_i^\top \;\simeq\; e_{g(i)} \otimes v_i, \tag{3}$$

so the payload of source i lands in row $g(i)$ by construction. A linear map of $[k_i; v_i]$ cannot produce this product. An additive memory has to build it with a nonlinear network and learn the matching read. We call this *implicit addressing*. It is possible with enough capacity, so our claims are about cost, not about what can be expressed.

Two realizations. The definition describes an interface rather than a specific layer. We use two realizations to make this explicit. **Cross-attention slots** use learned slot identities $A \in \mathbb{R}^{M\times p}$, with write and read projections A_{w} and A_{r}. Their write and read are

$$A^{\mathrm{in}} = \mathrm{softmax}_{\mathrm{src}}\left(\frac{A_{\mathrm{w}} R_{\mathcal{S}}^\top}{\tau_{\mathrm{in}}\sqrt{p}}\right), \qquad S = S^{(0)} + A^{\mathrm{in}} H_{\mathcal{S}} W_V W_O, \tag{4}$$

$$A^{\mathrm{out}} = \mathrm{softmax}_{\mathrm{slots}}\left(\frac{R_{\mathcal{T}} A_{\mathrm{r}}^\top}{\tau_{\mathrm{out}}\sqrt{p}}\right), \qquad O_{\mathcal{T}} = A^{\mathrm{out}} S U_V. \tag{5}$$

Thus the same indexed axis is used on both sides: it is an explicit *address axis* for organizing incoming payloads and selecting outgoing content. This is closely related to induced-set and Perceiver-style latent arrays (Lee et al., 2019; Jaegle et al., 2021).

Bilinear memories use the linear-attention fast-weight state (Katharopoulos et al., 2020),

$$\widetilde{S} = \sum_i \phi^{\mathrm{w}}(k_i) v_i^\top, \qquad o_j = \frac{\phi^{\mathrm{r}}(q_j)^\top \widetilde{S}}{\phi^{\mathrm{r}}(q_j)^\top Z}, \quad \phi(x) = \mathrm{ELU}(Wx + b) + 1. \tag{6}$$

There is no explicit slot object, but the M feature channels form the same addressable row axis. In both realizations, compact address codes select rows before the payload is decoded; the difference is whether the rows are explicit latents or feature channels of a bilinear state.

Reading into the node state. Addressability determines which memory content is retrieved; the retrieved content must still affect the target update. In both realizations we use the same small FiLM integration,

$$[\gamma_j, \beta_j] = \mathrm{MLP}_{\mathrm{film}}([h_j, o_j]), \qquad h_j^+ = (1 + \gamma_j) \odot h_j + \beta_j, \tag{7}$$

initialized near the identity. This keeps the comparison focused on the memory interface: source order is removed by the write, target order is preserved by the read, and the addressable memory only organizes distinctions already present in the node states.

3.2 Addressing from margin amplification

As in the constructive Two-Radius setting, split the sources into groups $I_m = \{i : g(i) = m\}$; each source carries the code of its group and each target the code of its own group.

Lemma 3.2 (Compact margin codes)**.** *For every $\varepsilon \in (0, 1)$ and $M \geq 2$, there are unit vectors $a_1, \ldots, a_M \in \mathbb{R}^p$ with $|\langle a_m, a_{m'} \rangle| < \varepsilon$ for all $m \neq m'$ as soon as $p > 4 \ln M / \varepsilon^2$. So M codes with constant margin $\Delta = 1 - \varepsilon$ exist in $p = O(\log M)$ dimensions.*

Random $\pm 1/\sqrt{p}$ codes work: each inner product is a sum of p bounded independent terms, so Hoeffding's inequality and a union bound over the M^2 pairs give the claim.

Lemma 3.3 (Affine one-hot addressing needs many dimensions)**.** *If $\phi(k) = Wk + b$ with $W \in \mathbb{R}^{M \times p}$, the response matrix $\Phi_{gm} = \phi_m(a_g)$ has rank at most $p + 1$. Exact one-hot gates, $\Phi = I_M$, therefore require $p \geq M - 1$.*

Codes alone only give a constant gap: the matching score is 1 and the others are at most $1 - \Delta$. An elementwise nonlinearity can turn this gap into a large ratio. We measure it by the *amplification ratio* of a nonnegative, nondecreasing σ with $\sigma(1) > 0$,

$$\kappa_\sigma(\Delta) = \sup_{x \leq 1 - \Delta} \sigma(x) \,/\, \sigma(1). \tag{8}$$

Theorem 3.4 (Addressing from margin amplification)**.** *Let $a_1, \ldots, a_M$ be unit codes with margin Δ, and $\phi_m^{\mathrm{w}}(k) = \phi_m^{\mathrm{r}}(k) = \sigma(\langle a_m, k \rangle)$ with $\kappa = \kappa_\sigma(\Delta)$. (i) With row normalization, the mass that row m receives from outside its group and the read weight of a target of group g on its own row satisfy*

$$\delta_m \leq \frac{n - |I_m|}{|I_m|}\, \kappa, \qquad \rho_{gg} \geq \frac{1}{1 + (M - 1)\kappa}. \tag{9}$$

(ii) With joint normalization, the weight that a target of group g puts on sources outside I_g satisfies

$$\varepsilon_g \leq \frac{n - |I_g|}{|I_g|} \big(2\kappa + (M - 2)\kappa^2\big). \tag{10}$$

Proof sketch. Rescale σ so that $\sigma(1) = 1$; both normalizations are invariant to this. With row normalization, row m gives weight 1 to each of the $|I_m|$ sources of its group and at most κ to each outsider, so the outside share is at most $(n - |I_m|)\kappa / |I_m|$; a target of group g gives weight 1 to its row and at most κ to each of the others. With joint normalization, the weight of source i is the kernel $\sum_m \sigma(\langle a_m, a_g \rangle) \sigma(\langle a_m, a_{g(i)} \rangle)$: it is at least 1 inside the group and, for an outsider, at most κ for $m = g$, κ for $m = g(i)$ and κ^2 for the others. Full proofs are in Appendix B.

Corollary 3.5 (Softmax, ELU $+1$ and threshold memories)**.** *With the codes of Lemma 3.2, $p = O(\log M)$ address dimensions and M rows suffice for (a) cross-attention slots, where $\sigma(x) = e^{x/\tau}$ gives $\kappa = e^{-\Delta/\tau}$; (b)* ELU $+1$ *bilinear memories whose mth pre-activation is $\beta(\langle a_m, k\rangle - 1) \le 0$, which stays on the exponential branch and gives $\kappa = e^{-\beta\Delta}$; and (c) thresholded maps such as $\sigma(x) = \max(0, x - 1 + \Delta/2)$, for which $\kappa = 0$. Leakage at most η needs $1/\tau$ or β of order $\Delta^{-1}\log(n/(\eta \min_m |I_m|))$.*

Softmax competition is therefore one way to build an addressable memory, not the only one. What matters is a small off-target/on-target ratio together with addressable multiplicative writes and reads. The $O(\log M)$ term refers only to the address code: both realizations still keep an $M \times d$ dynamic payload state. The condition on ELU $+1$ is also substantive. The construction in Corollary 3.5 keeps the relevant pre-activations on its negative, exponential branch; a generic use of linear attention does not automatically satisfy this margin-amplification condition. Thus the result identifies a sufficient memory interface rather than claiming that every attention or linear-attention layer is compactly addressable. Full proofs and additional rank statements are in Appendix B.

3.3 Capacity, and what an additive memory pays

Rows that can be selected separately count separately, so Theorem 2.1 applies with $L_{\text{eff}} = M$, and a matching construction exists.

Theorem 3.6 (Addressable construction)**.** *For $M \le n$, a permutation-invariant multiplicative write with M rows and a permutation-equivariant decoder solve Two-Radius with*

$$d \le \left\lceil \tfrac{n}{M} \right\rceil \left\lceil \tfrac{\log_2 n}{b} \right\rceil + \left\lceil \tfrac{\log_2 M}{b} \right\rceil + O(1) \tag{11}$$

coordinates per row, for both row- and jointly normalized memories.

Source i writes the code of $\pi(i)$ into block $r(i)$ of row $g(i)$, and target i reads the same row and block. This is a constructive bound; it does not say that training finds this partition.

Proposition 3.7 (Linear read rank)**.** *For a memory $S \in \mathbb{R}^{M\times d}$ read linearly by n targets as $O = RS$, $\operatorname{rank}(O) \le M$. Exact recovery of n arbitrary payloads of width $d \ge n$ therefore requires $M \ge n$.*

When $n > M$, some rows must store several payloads, and selecting one of them requires additional target-dependent decoding. The row axis reduces the routing problem; it does not remove the finite rank of a linear read.

An additive VN can run the same construction: ReLU networks compute margin gates and multiply them with payloads. The difference is the price. We make it precise in the idealized setting above, with one source per group ($n = M$); node states carry their code and, for sources, their payload.

Proposition 3.8 (Additive emulation and its cost)**.** *Let an additive memory write $z = b + \sum_g L\,\xi(h_g)$, that is $\psi = L\,\xi$ in Definition 3.1, with ξ continuous and $L \in \mathbb{R}^{D\times W}$ its last linear layer, and read $F(h_j, z) = G(Az + Bh_j + c)$ with first decoder layer $A \in \mathbb{R}^{W'\times D}$. (i)* Emulation. *If $\|v_g\|_\infty \le C$, ReLU networks ξ and G with two hidden layers and widths $O(M+Md)$ recover every payload exactly, with address codes of dimension $p = O(\log M)$. (ii)* Cost. *If the memory recovers every payload exactly for all payloads in U^M, with $U \subset \mathbb{R}^d$ open, then $D \ge Md$, $W \ge Md$ and $W' \ge Md$: the last write layer and the first read layer each have at least $(Md)^2$ weights. A multiplicative realization with shared projections uses $O(Mp + hp + p^2 + hd + d^2)$ parameters, where h is the node width.*

The lower bound is specific to the dense additive factorization stated in the proposition. On the write side, the target states $e_g \otimes v$ span all Md directions, so the final dense map must have rank at least Md. On the read side, the first dense map acts on the full state before the target-dependent nonlinearity can select a block; retaining arbitrary continuous payloads again requires an Md-dimensional bottleneck. A multiplicative memory performs the address–payload product before pooling and selects along the memory axis before the small payload decoder. The result therefore formalizes one cost of implicit addressing for ordinary dense MLP interfaces; it is not a lower bound on every possible structured VN implementation.

Scope and trade-offs. Addressability does not remove finite-memory limits. A decoder that is linear in the memory contents can recover at most M arbitrary payload directions, so packing

more than M entries into M rows requires additional conditional structure in the target decoder (Proposition 3.7). Compact addresses also rely on a non-vanishing margin. At fixed p, the number of well-separated codes is finite; increasing M must eventually reduce the available margin or require a larger p. Sharper amplification can compensate in the bounds, but also changes the optimization landscape. At a fixed state budget Md, more rows additionally imply narrower payloads. These trade-offs are the reason the logarithmic address dimension should not be read as logarithmic memory capacity.

The same distinction is visible in a degenerate routing regime. If the write scores are independent of the source, every row receives the same pooled source-dependent component and the effective factorization falls back to one, even if the rows have different learned initial states. In practice this can be monitored through write entropy and read usage. The guarantees above are representational: row-normalized slots and jointly normalized bilinear memories can satisfy the same addressing bounds while exhibiting different training dynamics. Appendix B gives the corresponding details.

4 Preserving Query Support

Addressability decides *where* information is stored. A second question is how strongly the sources support a read. Let $w(q, x) \geq 0$ be the weight that query or row q gives to source x, and

$$Z(q, X) = \sum_{x \in X} w(q, x) \tag{12}$$

the *support* of the read. It is not just a count: each query learns its own notion of a match.

Proposition 4.1 (Normalized reads are blind to replication)**.** *For a row-normalized or jointly normalized read, replicating every source r times leaves the read unchanged, while $Z(q, rX) = rZ(q, X)$. An unnormalized read scales with r.*

Numerator and denominator grow by the same factor. Thus, when the public interface exposes only the normalized read, uniform replication is not observable through that read. This includes row-normalized slots and normalized linear-attention reads unless they expose an additional denominator or mass state. An unnormalized sum retains the scale but loses the bounded convex-read property. Self-attention often has an internal reference because the query token also contributes its own key; a latent source-to-memory query need not have an analogous element.

A private reference. For each row (and head), we add one reference with weight $w_0 = e^{s_0}$ and value a that is not a source:

$$z(X) = \frac{w_0\, a + \sum_i w_i v_i}{w_0 + \sum_i w_i}, \qquad Z = w_0\, \frac{1 - \alpha_0}{\alpha_0}, \quad \alpha_0 = \frac{w_0}{w_0 + Z}. \tag{13}$$

The normalized content $\bar{v} = \sum_i (w_i/Z) v_i$ and the support relative to the reference are then both available. We learn s_0 per row/head and feed the log-odds $\log Z - s_0$ to a small MLP that bypasses content normalization. For a jointly normalized memory, the same construction adds a positive reference vector c_0 to the denominator and $c_0 \odot a$ to the numerator. If an implementation already exposes its raw denominator accumulator, that quantity itself provides a mass path and the reference is not needed for observability.

Proposition 4.2 (Anchored reads are bounded and support-aware)**.** *The read of Eq.* (13) *is a convex combination of a and the source values, so $\|z(X)\| \leq \max(\|a\|, \max_i \|v_i\|)$ for any number of sources. Its log-odds $\log Z - s_0$ are strictly increasing in Z.*

With a zero anchor value,

$$z(X) = \frac{Z}{e^{s_0} + Z}\, \bar{v},$$

so s_0 sets the support scale at which the content becomes active. Softmax$_1$ (Miller, 2023) is the fixed case $s_0 = 0$. In self-attention it lets a head put little mass on real tokens; here the reference instead supplies the "self" term that a latent cross-attention query does not naturally have, and its weight provides the support signal.

Theorem 4.3 (Cardinality as a special case)**.** *On multisets of bounded size over a finite color alphabet, anchored attention followed by an expressive map represents X injectively, and can therefore implement one 1-WL refinement on bounded colored neighborhoods. A normalized read without a separate mass path is not injective on any domain containing X and rX, $r > 1$.*

With zero source logits, one-hot color values and a zero-logit reference, the reference weight is $1/(1 + |X|)$ and the other coordinates hold the color histogram scaled by $|X|/(1 + |X|)$, which recovers both (Appendix D).

5 EXPERIMENTS

We test whether making the global shortcut explicitly addressable helps when oversquashing is controlled: Two-Radius fixes distance while the number of associations grows, whereas Tree-NeighborsMatch increases both. On these tasks we evaluate two addressable realizations, cross-attention slots and the anchored bilinear memory. We then use permutation-invariant semantic routing to test content-based addressing without persistent identifiers, and close with a short summary of standard LRGB benchmarks. Protocols are in the appendix.

5.1 TWO-RADIUS

Addressability. On identifier-based Two-Radius with a 192-scalar memory, we remove the two halves of the address interface in turn (Table 1). Full slots, with routed writes and target-to-slot reads, are exact on all seeds within 500 updates. Keeping routed writes but reading the flattened memory with a target-conditioned MLP eventually reaches 94.3%, with $8\times$ more parameters, and only two of three seeds recover full graphs. Without routed writes, the rows receive the same summary and accuracy stays below 60%. A strong implicit decoder can thus read a routed memory, as Proposition 3.8 allows, but in this experiment it requires a larger decoder and shows greater variability across seeds (Appendix F).

Table 1: Two-Radius write/read ablation: target accuracy (%) after a given number of updates, mean $\pm$ s.d. over three seeds.

Model	Params	500	2k	5k
No routed write	3.85k	10.7 ± 0.5	49.0 ± 12.3	58.6 ± 13.5
Routed write, implicit read	32.2k	79.2 ± 2.7	93.9 ± 10.6	94.3 ± 9.8
Full routed write/read	3.85k	$\mathbf{100.0}$	$\mathbf{100.0}$	$\mathbf{100.0}$

Support. In multiplicity-aware Two-Radius, each identifier i has a label $\pi(i)$ and a base multiplicity $a_i \in \{1, \dots, 4\}$; each assignment is rendered at scales $r \in \{1, 2, 3\}$, so source $(i, \pi(i))$ appears $c_i = ra_i$ times, and target t_i predicts $(\pi(i), c_i)$. The label tests addressing; the count tests support.

Table 2 separates the two mechanisms in a controlled run. Normalized slots recover the label associations but not the common replication scale, consistent with Proposition 4.1. Both anchored realizations recover the label and count outputs in this seed, showing that the reference mechanism is not specific to the row-normalized slot implementation. We use this experiment as a mechanistic test of the information exposed by the normalization interface, not as a general performance comparison.

5.2 TREE-NEIGHBORSMATCH

We follow Tree-NeighborsMatch (Alon & Yahav, 2021): candidate leaves carry labels and distinct local neighbor-count features, while the root receives one such count and must predict the label of the matching candidate. As r grows, 2^r candidate associations must reach the root. The benchmark supplies the count feature, so the comparison focuses on communication. We evaluate a local MPNN, pooled homogeneous VNs, and addressable slot memories under the same data generator. We also report a separate diagnostic for the standard persistent OGB-style virtual node in Appendix H.

Table 3 shows the clearest separation at $r = 5$. The local MPNN reaches $3.1 \pm 0.2\%$, and the two pooled VN variants reach about 10.6%, whereas the slot variants remain near 100% with substantially

Table 2: Multiplicity-aware Two-Radius, validation accuracy (%), single seed.

Model	Label	Count	Both
MPNN	7.8	17.1	1.3
Additive VN	8.9	18.3	1.4
Normalized slots	100.0	28.3	28.3
Anchored slots	**100.0**	**100.0**	**100.0**
Anchored bilinear memory	**100.0**	**100.0**	**100.0**

Table 3: Tree-NeighborsMatch test accuracy (%) by tree depth r (2^r candidates/classes; chance is 2^{-r}). Mean $\pm$ s.d. over three seeds. Parameters are reported at $r = 5$.

Model	Params	$r = 3$	$r = 4$	$r = 5$	$r = 6$
MPNN ($r + 1$ layers)	92k	99.6 ± 0.7	16.9 ± 0.9	3.1 ± 0.2	5.1 ± 0.2
Pooled VN	363k	100.0	73.9 ± 0.2	10.6 ± 0.1	5.3 ± 0.2
Wide pooled VN	2.33M	100.0	75.2 ± 5.5	10.7 ± 0.3	5.7 ± 0.2
Slots	37.5k	**100.0**	**100.0**	**100.0**	$\mathbf{26.2 \pm 0.2}$
Anchored slots	38.9k	**100.0**	**100.0**	99.9 ± 0.1	25.7 ± 0.2

fewer parameters. The same pattern continues at $r = 6$: slots reach $26.2 \pm 0.2\%$ and anchored slots $25.7 \pm 0.2\%$, while the pooled baselines remain between 5.1% and 5.7%. An anchored bilinear memory gives a second addressable realization, reaching 100% at $r = 3, 4, 5$ and a mean of 20.9% at $r = 6$. The reference has little effect on the slot results here, as expected for a task that tests addressability rather than support scale. A separate persistent OGB-style VN check is reported in Appendix H.

5.3 Permutation-invariant semantic routing

Each graph is a fresh episode with eight sources and eight targets. Sources carry a continuous key and a categorical payload; each target receives a perturbed copy of one key and must predict its payload. Keys are resampled for every graph, and there are no identifiers or orderings, so the address has to be inferred from content in each episode. All models have exactly 192 dynamic memory scalars and are trained for 50k updates.

The *parameter-matched VN* keeps these 192 scalars in one homogeneous state and uses a small width-5, depth-2 nonlinear write/read path (2.29k parameters). The *high-capacity VN* keeps the same 192-dimensional state but gives implicit addressing much more capacity: three-layer nonlinear source writing and three-layer target-conditioned decoding with hidden width 96 (58.95k parameters). The addressable slot model instead uses 32 rows of width 6 and a shared semantic projection for source addresses and target queries (2.09k parameters). Each family is tuned separately; further optimization details are in Appendix E.

Because the keys are resampled in every episode, the VN cannot memorize a fixed semantic-type-to-coordinate convention. Any implicit addressable organization must be created from the current source set and decoded again from the target query. The slot model keeps a stable addressable row axis while allowing the contents assigned to those rows to change from episode to episode.

Table 4 compares the three models at the same 192-scalar dynamic-memory budget. The parameter-matched VN remains near its low-accuracy solution under this training budget. The high-capacity VN shows that a homogeneous state can learn strong implicit addressing when its writer and target-conditioned decoder are made much larger. It reaches 93.9% IID accuracy, but uses $28\times$ more parameters and about $22\times$ more dense compute than the explicit addressable slot model, and the gap is larger under stronger key noise. Proposition 3.8 gives a scale comparison for continuous payloads, but it is not a lower bound for this categorical benchmark. Here the comparison asks a narrower empirical question: how much capacity is needed when the addressable memory axis is learned implicitly rather than exposed by the architecture.

Table 4: Semantic routing with 192 memory scalars. Mean ± s.d. over three seeds at 50k updates. OOD uses stronger key noise.

Model	Params	MFLOPs	IID target	OOD target	OOD exact
Parameter-matched VN	2.29k	0.033	30.7 ± 0.3	30.7 ± 0.3	0.0
High-capacity VN	58.95k	0.934	93.9 ± 1.0	74.5 ± 2.6	9.6 ± 2.6
Cross-attention slots	2.09k	0.042	$\mathbf{96.4 \pm 0.1}$	$\mathbf{84.8 \pm 0.3}$	$\mathbf{25.4 \pm 1.5}$

5.4 Standard benchmarks

We finally use three LRGB tasks (Dwivedi et al., 2022) as a transfer check: Peptides-func, Peptides-struct, and PascalVOC-SP. A light, non-persistent anchored slot block on top of an OGB-style VN gives similar five-seed mean scores on Peptides-func and PascalVOC-SP and a lower mean MAE on Peptides-struct. The anchored bilinear realization gives worse mean scores on these tasks (0.680 AP, 0.247 MAE, and 0.389 F1), despite matching the slot memory on the controlled addressability benchmarks. The two addressable parameterizations therefore behave similarly on the mechanism tests but not on every downstream task. Full means and protocol details are in Appendix I.

6 Related Work

Oversquashing and virtual nodes. Oversquashing has been studied through receptive-field growth, curvature and communication bottlenecks (Alon & Yahav, 2021; Topping et al., 2022; Di Giovanni et al., 2023; Loukas, 2020); Two-Radius isolates its capacity part (Mishayev et al., 2025). Virtual nodes give a cheap global shortcut (Gilmer et al., 2017; Ishiguro et al., 2019); other work studies their mixing (Southern et al., 2025; Hwang et al., 2022), their relation to attention and expressivity (Cai et al., 2023; Rosenbluth et al., 2024), and relays through attention nodes (Caruso et al., 2026). We study how the memory of the shortcut should be organized once it exists.

Latent arrays, linear attention and attention support. Our row-normalized realization is close to induced set attention and Perceiver latents (Lee et al., 2019; Jaegle et al., 2021); the jointly normalized one is the fast-weight state of linear attention (Katharopoulos et al., 2020), whose feature dimension bounds what a linear decoder can recover (Schlag et al., 2021). Softmax$_1$ adds a fixed unit reference to the softmax (Miller, 2023), and gated attention lets heads abstain (Bondarenko et al., 2023); registers and attention sinks also add special attention elements (Darcet et al., 2024; Xiao et al., 2024). Injective multiset aggregation underlies 1-WL expressivity (Xu et al., 2019; Morris et al., 2019); PNA, cardinality-preserving attention and ACAM keep counts explicitly (Corso et al., 2020; Zhang & Xie, 2020; Marcoccia et al., 2026). Here cardinality is one special case of observable support.

7 Discussion and Conclusion

A virtual node removes long communication paths, but its shared state can still become a bottleneck. We identify a simple route to an *addressable* global memory: separated address codes, margin amplification, and multiplicative write/read maps create an explicit address axis, with $O(\log M)$ address-code dimensions under a constant-margin assumption. Cross-attention slots and the studied $\mathrm{ELU}+1$ bilinear memory are two realizations.

Normalized cross-attention raises a separate issue because a latent query has no natural self-key reference. A learned private anchor supplies that reference and exposes query support while keeping the read bounded. The controlled Two-Radius and Tree-NeighborsMatch experiments support explicit addressability and support-aware reads, while LRGB shows that different addressable realizations need not behave identically on general downstream tasks, especially regarding optimization. This paves the way for further research to design powerful, addressable and support-aware virtual nodes.

Reproducibility Statement

Complete proofs are in Appendices B–D. Appendices E–H specify the synthetic generators, models, budgets and diagnostics, and Appendix I the real-graph protocol and all runs. Executable implementations and data generators will be submitted as anonymized supplementary material.

AI Use Statement

In this work, we used generative AI tools through prompt-accessible web interfaces to help draft and edit parts of the manuscript text, to propose and check parts of the mathematical derivations and proofs, and to write and debug experiment code. All experiments were run by the author, and generative AI tools were not used to generate, alter or select data or results. We have reviewed all AI-assisted work: the derivations were checked by the author and the experiment code was tested before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI. No autonomous agentic AI tool was used for this work.

References

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between MPNN and graph transformer. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3408–3430, 2023.

Alessandro Caruso, Jacopo Venturin, Lorenzo Giambagli, Edoardo Rolando, Zakariya El-Machachi, Frank Noé, and Cecilia Clementi. Extending the range of graph neural networks with global encodings. *Nature Communications*, 17:1855, 2026. doi: 10.1038/s41467-026-69715-3.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Liò, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7865–7885, 2023.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. An analysis of virtual nodes in graph neural networks for link prediction. In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=dI6KBKNRp7. Extended abstract.

Katsuhiko Ishiguro, Shin-ichi Maeda, and Masanori Koyama. Graph warp module: An auxiliary module for boosting the power of graph neural networks in molecular graph analysis, 2019.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, 2020.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, 2019.

Andreas Loukas. What graph neural networks cannot learn: Depth vs width. In *International Conference on Learning Representations*, 2020.

Félix Marcoccia, Victor Fagoo, Gilles Monzat de Saint Julien, Cédric Adjih, Thomas Watteyne, and Paul Mühlethaler. NetDiff: Graph diffusion with improved global capabilities to generate and update mobile network topologies. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. Accepted at ICML 2026; proceedings forthcoming.

Evan Miller. Attention is off by one. `https://www.evanmiller.org/attention-is-off-by-one.html`, July 2023. Accessed 2026-09-16.

Yaaqov Mishayev, Yonatan Sverdlov, Tal Amir, and Nadav Dym. Short-range oversquashing, 2025. Accepted at the Fourth Learning on Graphs Conference (LoG 2025); proceedings forthcoming.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019. doi: 10.1609/aaai.v33i01.33014602.

Eran Rosenbluth, Jan Tönshoff, Martin Ritzert, Berke Kisin, and Martin Grohe. Distinguished in uniform: Self-attention vs. virtual nodes. In *International Conference on Learning Representations*, 2024.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, 2021.

Joshua Southern, Francesco Di Giovanni, Michael Bronstein, and Johannes F. Lutzeyer. Understanding virtual nodes: Oversquashing and node heterogeneity. In *International Conference on Learning Representations*, 2025.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *arXiv preprint arXiv:2309.00367*, 2023.

Jake Topping, Francesco Di Giovanni, Benjamin P. Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=7UmjRGzp-A`.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Shuo Zhang and Lei Xie. Improving attention mechanism in graph neural networks via cardinality preservation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 1395–1402, 2020. doi: 10.24963/ijcai.2020/194.

A The Two Realizations in Detail

Cross-attention slots. Let $A = [a_1, \dots, a_M]^\top \in \mathbb{R}^{M\times p}$ be learned slot identities, with write and read projections $A_{\mathrm{w}} = q_{\mathrm{w}}(A)$ and $A_{\mathrm{r}} = k_{\mathrm{r}}(A)$, and let $R_{\mathcal{S}}$ and $R_{\mathcal{T}}$ stack the routing keys and queries. The write and the read are

$$A^{\mathrm{in}} = \mathrm{softmax}_{\mathrm{src}}\left(\frac{A_{\mathrm{w}} R_{\mathcal{S}}^\top}{\tau_{\mathrm{in}}\sqrt{p}}\right), \qquad S = S^{(0)} + \left(A^{\mathrm{in}} H_{\mathcal{S}} W_V\right) W_O, \tag{14}$$

$$A^{\mathrm{out}} = \mathrm{softmax}_{\mathrm{slots}}\left(\frac{R_{\mathcal{T}} A_{\mathrm{r}}^\top}{\tau_{\mathrm{out}}\sqrt{p}}\right), \qquad O_{\mathcal{T}} = A^{\mathrm{out}} S U_V. \tag{15}$$

The slot identity is the address and S_m is the payload stored at that address. The same identities generate write queries and read keys, so slot m is the same compartment on both sides.

Bilinear memory. The state is $\widetilde{S} = \sum_i \phi^{\mathrm{w}}(k_i) v_i^\top \in \mathbb{R}^{M\times d}$ with $\phi(x) = \mathrm{ELU}(Wx + b) + 1$ and M feature channels, and a target reads $\phi^{\mathrm{r}}(q_j)^\top \widetilde{S} \,/\, \phi^{\mathrm{r}}(q_j)^\top Z$. The input address has dimension p while the feature map has dimension M: the $O(\log M)$ statement concerns the former. Standard linear attention applies ϕ coordinatewise with $M = p$; its state then has only p rows, which caps the number of exactly recoverable payloads at p (Proposition 3.7), and on the affine branch of $\mathrm{ELU}+1$ overlaps between codes stay of constant order. The guarantee of Corollary 3.5(b) instead lifts p-dimensional keys to M features whose pre-activations stay on the exponential branch.

B Addressing Proofs

B.1 Proof of Lemma 3.2

Draw each coordinate of each a_m independently and uniformly from $\{\pm 1/\sqrt{p}\}$, so every a_m is a unit vector. For $m \neq m'$, $\langle a_m, a_{m'}\rangle$ is a sum of p independent variables in $[-1/p, 1/p]$ with zero mean, and Hoeffding's inequality gives

$$\Pr\left(|\langle a_m, a_{m'}\rangle| \geq \varepsilon\right) \leq 2\exp\big(-p\varepsilon^2/2\big).$$

A union bound over the $M(M-1)/2$ pairs bounds the probability that some pair fails by $M^2\exp(-p\varepsilon^2/2)$, which is smaller than one when $p > 4\ln M/\varepsilon^2$. Hence a valid codebook exists. □

B.2 Proof of Lemma 3.3

Stack the codes as rows of $C \in \mathbb{R}^{M\times p}$. The response matrix is $\Phi = CW^\top + \mathbf{1}b^\top$, a sum of a matrix of rank at most p and a matrix of rank at most one. Hence $\mathrm{rank}\,\Phi \leq p+1$, and $\Phi = I_M$ requires $p + 1 \geq M$. □

B.3 Proof of Theorem 3.4

Both normalizations are invariant to a positive rescaling of σ, so assume $\sigma(1) = 1$; then $\sigma(\langle a_m, a_{m'}\rangle) \leq \kappa$ for $m \neq m'$.

(i) Row normalization. Row m gives weight $\sigma(\langle a_m, a_{g(i)}\rangle)$ to source i: each of the $|I_m|$ group members contributes $\sigma(1) = 1$ and each outsider at most κ. The outside mass is therefore

$$\delta_m = \frac{\sum_{i\notin I_m}\sigma(\langle a_m, a_{g(i)}\rangle)}{\sum_i \sigma(\langle a_m, a_{g(i)}\rangle)} \leq \frac{(n - |I_m|)\kappa}{|I_m|}.$$

A target of group g gives row g the read weight $\rho_{gg} = \sigma(1)/\sum_m \sigma(\langle a_m, a_g\rangle) \geq 1/(1 + (M-1)\kappa)$.

(ii) Joint normalization. The read of a target with query a_g weights source i proportionally to $K_i = \sum_m \sigma(\langle a_m, a_g\rangle)\sigma(\langle a_m, a_{g(i)}\rangle)$. For $i \in I_g$, the term $m = g$ equals $\sigma(1)^2 = 1$, so $K_i \geq 1$. For $i \in I_h$ with $h \neq g$, the term $m = g$ is $\sigma(1)\sigma(\langle a_g, a_h\rangle) \leq \kappa$, the term $m = h$ is $\sigma(\langle a_h, a_g\rangle)\sigma(1) \leq \kappa$, and each of the $M-2$ remaining terms is at most κ^2. Hence $K_i \leq 2\kappa + (M-2)\kappa^2$, and

$$\varepsilon_g = \frac{\sum_{i\notin I_g} K_i}{\sum_i K_i} \leq \frac{(n - |I_g|)\big(2\kappa + (M-2)\kappa^2\big)}{|I_g|}. \qquad \square$$

B.4 Proof of Corollary 3.5

For $\sigma(x) = e^{x/\tau}$, $\sup_{x \le 1-\Delta} \sigma(x)/\sigma(1) = e^{-\Delta/\tau}$. For the bilinear memory, choose the rows of both W_k and W_q as βa_m and set $b_k = b_q = -\beta \mathbf{1}$. For unit address codes, the corresponding pre-activation is nonpositive, where $\mathrm{ELU}(x) + 1 = e^x$; hence $\sigma(x) = e^{\beta(x-1)}$ and $\kappa = e^{-\beta\Delta}$. For the thresholded map, $\sigma(x) = 0$ for $x \le 1 - \Delta$ and $\sigma(1) = \Delta/2 > 0$, so $\kappa = 0$. Combining with Lemma 3.2 gives $p = O(\log M)$. Finally, the bound of Eq. (9) is at most η as soon as $\kappa \le \eta \min_m |I_m|/n$, which for the exponential maps holds when $1/\tau$ or β is at least $\Delta^{-1} \log(n/(\eta \min_m |I_m|))$; the joint bound behaves in the same way up to a constant. □

B.5 Scope of the addressing results

What $O(\log M)$ means. The logarithm concerns the dimension p of the address code. It does not say that M compartments fit in $O(\log M)$ dynamic scalars: cross-attention uses M slots and the bilinear memory M feature channels, and both keep an $M \times d$ payload state, plus an M-dimensional normalizer for joint normalization. The gain is that a node does not need an M-dimensional one-hot identifier to select a row.

No entirely free lunch. The result assumes that the model learns M addresses with a constant margin Δ. For a fixed p, only a limited number of unit codes keep a given separation, so increasing M eventually shrinks the achievable margin. Keeping the bounds of Theorem 3.4 small then needs a larger p or a stronger amplification (lower τ, larger β), which makes routing sharper and potentially harder to optimize. Addresses also trade against payload: at a fixed budget Md, more rows means narrower rows, and each read returns a single mixture of rows. Theorem 3.6 makes this explicit, since each row must hold $\lceil n/M \rceil$ label blocks of $\lceil \log_2 n/b \rceil$ coordinates. At the other extreme, if the write scores are constant over sources for every row (for instance $q_{\mathrm{w}} \equiv 0$), every row receives the same pooled summary, $L_{\mathrm{eff}} = 1$, and the memory behaves like one mean-pooled VN (Lemma C.1). This regime is observable: the write entropy stays at its uniform value and few rows receive read mass.

Remark B.1 (Optimization scope). The addressing bounds are representational and do not imply identical training dynamics across row-normalized and jointly normalized realizations. The main text discusses this distinction; the bounds themselves concern only the existence and leakage of the address interface.

B.6 Proof of Proposition 3.7

For fixed targets, a decoder linear in the memory contents produces $O = R\,S$ with $S \in \mathbb{R}^{M \times d}$ the memory and $R \in \mathbb{R}^{n \times M}$ the read coefficients, so $\operatorname{rank} O \le M$. Arbitrary payloads of width $d \ge n$ can form a matrix of rank n, which requires $M \ge n$. □

B.7 Proof of Proposition 3.8

(i) Emulation. Take the codes of Lemma 3.2 with margin Δ. The gate of row m,

$$\gamma_m(k) = \mathrm{ReLU}(s_m) - \mathrm{ReLU}(s_m - 1), \qquad s_m = \tfrac{2}{\Delta}\big(\langle a_m, k\rangle - 1 + \tfrac{\Delta}{2}\big),$$

equals 1 when $k = a_m$ (then $s_m = 1$) and 0 when $\langle a_m, k\rangle \le 1 - \Delta$ (then $s_m \le -1$). It uses two ReLU units per row. For a scalar x with $|x| \le C$ and a gate $\gamma \in \{0, 1\}$,

$$\gamma x = \mathrm{ReLU}\big(x - C(1-\gamma)\big) - \mathrm{ReLU}\big(-x - C(1-\gamma)\big),$$

since the right-hand side is x when $\gamma = 1$ and 0 when $\gamma = 0$.

Write. The first hidden layer of ξ computes the $2M$ gate units from the code and passes the payload through $2d$ units ($x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x)$). The second hidden layer applies the product identity to every pair (row m, coordinate c), with $2Md$ units. The last layer L sums each pair, so $L\,\xi(h_g) = e_g \otimes v_g$ and $z = b + \sum_g e_g \otimes v_g$ holds every payload in its own block.

Read. The first layer of the decoder computes the $2M$ gate units from the target code (through Bh_j) and passes the Md coordinates of $z - b$ through $2Md$ units (through Az). The second hidden layer applies the product identity to every pair (m, c) with the target's gates, and the output layer sums

over m, which returns $v_{g(j)}$. Widths are $O(M + Md)$ on both sides, and the codes have dimension $p = O(\log M)$.

(ii) Cost. We use invariance of domain: there is no continuous injective map from a nonempty open subset of $\mathbb{R}^N$ to $\mathbb{R}^{N'}$ with $N' < N$.

The target state does not depend on the payloads, so every target reads its payload from z alone. Since each group has its own target, z determines all payloads, and the map $(v_1, \dots, v_M) \mapsto z$ is injective on U^M. It is continuous because ξ is. Hence $D \geq Md$. Moreover $z - b$ lies in the range of L, a subspace of dimension at most $\operatorname{rank} L \leq W$. The same argument gives $W \geq Md$, so L has $DW \geq (Md)^2$ weights.

On the read side, the output of target j depends on z only through Az, because $Bh_j + c$ does not depend on the payloads. Each target recovers its own payload from Az, and every group has a target, so all payloads are determined by Az. The map $(v_1, \dots, v_M) \mapsto Az$ is therefore continuous and injective on U^M, which gives $W' \geq Md$ and at least $W'D \geq (Md)^2$ weights in A.

For the multiplicative count, the learned address table contributes $O(Mp)$ parameters; shared node/address projections contribute $O(hp + p^2)$; and the value/output maps contribute $O(hd + d^2)$. None of the dense shared maps has a width proportional to the product Md. □

C Capacity Proofs

C.1 Proof of Theorem 2.1

For each permutation $\pi \in S_n$, let T_π be the full source-dependent transcript available to the targets. If $\pi \neq \pi'$ but $T_\pi = T_{\pi'}$, fixed target identifiers and a deterministic decoder produce the same joint output on both instances, which contradicts exact recovery. Therefore $\pi \mapsto T_\pi$ must be injective, so the transcript must take at least $n!$ distinct values.

With L_{eff} independently usable states, d coordinates per state, and at most 2^b values per coordinate, the transcript has at most $2^{L_{\text{eff}} db}$ possible values. Hence $2^{L_{\text{eff}} db} \geq n!$, which proves Eq. (1). Stirling's approximation gives $\log_2(n!) = n \log_2 n - O(n)$. With several global layers, $L_{\text{eff}} db$ should be replaced by the capacity of the complete source-dependent transcript. For a multiplicative memory whose M rows can vary independently and are separately selectable, $L_{\text{eff}} = M$. Average-case bounds would require a rate–distortion argument.

C.2 Why symmetric virtual nodes do not factorize memory

Lemma C.1 (Symmetric VN clones do not factorize memory)**.** *Consider M VNs with identical initial states, identical neighborhoods, and shared permutation-equivariant update functions. Their states remain identical at every layer.*

Proof. The property holds at initialization. If all VNs are identical at layer ℓ, they receive identical neighbor multisets and aggregated messages. The shared update therefore produces identical states at layer $\ell + 1$. □

Their joint state always has the form $(z, \dots, z)$, so it has no more reachable values than one VN and contributes only one effective compartment.

C.3 Proof of Theorem 3.6

Let $k = \lceil n/M \rceil$ and fix a public partition $I_1, \dots, I_M$ with $|I_m| \leq k$. Let $g(i)$ and $r(i)$ denote the group and within-group position of identifier i. A label in $[n]$ uses $q = \lceil \log_2 n/b \rceil$ coordinates of b bits. Row m stores its $\lceil \log_2 M/b \rceil$-coordinate group address and k label blocks of length q. Source i uses the address of row $g(i)$ and a payload v_i that contains the code of $\pi(i)$ in block $r(i)$ and zeros elsewhere. In the hard-routing limit of Theorem 3.4 ($\kappa \to 0$, or exactly with a thresholded map), a row-normalized memory stores in row m the average of its group's payloads, and a jointly normalized memory returns the same average to a target of group m. Blocks do not overlap within a group, so each block holds one label code scaled by $1/|I_m|$, a public constant that the decoder inverts. Target j

reads row $g(j)$ and block $r(j)$, recovering $\pi(j)$. Source order does not matter and permuting targets only permutes their reads, which proves equivariance and Eq. (11).

For labels independently drawn from an alphabet of size C, the analogous capacity requirement is $L_{\text{eff}} db \geq n \log_2 C$, and the construction uses $O((n/M) \log C/b)$ label coordinates per row.

D SUPPORT-MASS PROOFS

Proposition 4.1. Under uniform replication, every source weight and value appears r times. For a row-normalized read, both the weighted value sum and the row normalizer scale by r, so the normalized row content is unchanged. For a jointly normalized read, numerator and denominator also scale by the same factor. Hence the exposed normalized read is unchanged while Z scales by r. An unnormalized read scales with r. □

Proposition 4.2. The weights $w_0/(w_0 + Z)$ and $w_i/(w_0 + Z)$ are nonnegative and sum to one, so $z(X)$ lies in the convex hull of $\{a\} \cup \{v_i\}$ and its norm is at most $\max(\|a\|, \max_i \|v_i\|)$. The log-odds equal $\log(Z/w_0) = \log Z - s_0$, strictly increasing in Z, and $Z(rX) = rZ(X) > Z(X)$ for $r > 1$. □

Theorem 4.3. Let the alphabet be $\{1, \ldots, C\}$, use one-hot values $v_i = e_{c(i)}$, zero source logits, and an anchor with zero logit and value e_0 on a reserved coordinate. Then $z_0 = 1/(1 + |X|)$ and $z_c = n_c/(1 + |X|)$, where n_c is the number of elements of color c. The map $X \mapsto z$ is injective: $|X|$ follows from z_0, and each $n_c = z_c/z_0$. A sufficiently expressive map applied to z and to the center color then implements one 1-WL refinement on neighborhoods of bounded size. For a normalized memory without a separate cardinality path, Proposition 4.1 gives identical reads on X and rX, so it cannot be injective on a domain containing both. □

Scope of Proposition 4.1. The proposition does not say that a system can never retain Z. A linear-attention implementation may keep its denominator accumulator, and an architecture that exports it already has a mass path. The question is how to keep a bounded normalized read while making the support available through the same interface. For a jointly normalized memory, the reference is implemented by adding a positive vector c_0 to the denominator and $c_0 \odot a$ to the numerator; if the raw denominator is already exposed to the decoder, this is unnecessary for observability. With $a = 0$, the anchored read is $z(X) = \frac{Z}{e^{s_0} + Z} \bar{v}$: the normalized content is multiplied by a support gate, and s_0 sets the scale at which each row or head becomes active.

Relation to Softmax$_1$. With zero anchor value, $z_h(X) = \frac{Z_h}{e^{s_{0,h}} + Z_h} \bar{v}_h$. Setting $s_{0,h} = 0$ gives Softmax$_1$ exactly. The fixed $+1$ already breaks replication invariance; the learned anchor calibrates the reference separately for each row and head, and the log-odds path exposes it explicitly.

E PERMUTATION-INVARIANT SEMANTIC ROUTING PROTOCOL

Episodes. Each episode contains $n = 8$ source key–payload pairs and eight target queries. New base keys are sampled for every graph as normalized random vectors in $\mathbb{R}^8$. A source key and its matching target query are independent Gaussian perturbations of the same base key followed by ℓ_2 normalization. Payloads are sampled independently from eight classes. Source and target orders are independently permuted. Train/IID key noise is 0.07 and the main OOD split is 0.16.

Best-tuned 192-scalar comparison. All models contain exactly 192 dynamic global-memory scalars and are trained for $50{,}000$ updates with batch size 192. The slot model uses 32 slots of width 6, hidden width 40, temperature 0.14, and a semantic projection shared between source keys and target queries. Its learning rate is stepped from 3×10^{-2} to 10^{-2}, 3×10^{-3}, and 10^{-3} over training. It contains $2{,}094$ learned parameters.

For the parameter-matched VN, the AdamW learning rate is 1.5×10^{-3} through 10k updates, 7.5×10^{-4} through 30k, and 3×10^{-4} thereafter. Each family is tuned separately; final evaluation uses seeds 100–102.

Approximate forward FLOPs. A multiply plus an add counts as two FLOPs; biases, activations, normalization, and softmax are omitted. The parameter-matched VN uses 0.0333 MFLOPs, the high-capacity VN 0.934 MFLOPs, and the 32×6 slot model 0.0420 MFLOPs per graph under this convention.

Post-hoc noise robustness. For representative seed-100 models frozen after 50k updates, target accuracy (%) is:

Noise σ	High-capacity VN	Slots
0.07	94.6	96.3
0.16	75.7	84.8
0.20	66.6	77.0
0.24	59.4	69.1
0.30	51.4	58.4

This is a benchmark-specific robustness pattern, not a general OOD guarantee.

Permutation diagnostic. Reordering only the source set changes slot-model logits by at most 1.9×10^{-6}, consistent with source-order invariance up to floating-point error.

F Long-Budget Write/Read Ablation

We rerun the identifier-based Two-Radius lookup with online data for $5{,}000$ updates and three seeds. The slot memory has $16 \times 12 = 192$ dynamic scalars. The No routed write variant gives every slot the same pooled source-dependent summary but keeps target-to-slot attention. The Implicit read variant keeps content-dependent routed writes, flattens the whole slot memory, and uses a target-conditioned MLP decoder. The Full slots model uses routed writes and target-to-slot reads.

Table 5: Long-budget Two-Radius write/read ablation. Target accuracy (%) at fixed update budgets; mean $\pm$ s.d. over three seeds.

Model	Params	500	1k	2k	5k
No routed write	3.85k	10.7 ± 0.5	23.4 ± 2.3	49.0 ± 12.3	58.6 ± 13.5
Routed write, implicit read	32.2k	79.2 ± 2.7	93.1 ± 12.0	93.9 ± 10.6	94.3 ± 9.8
Full routed write/read	3.85k	100.0	100.0	100.0	100.0

At 5k updates, two of three implicit-read seeds achieve exact graph recovery and one remains near 83% target accuracy, giving $66.7 \pm 57.7\%$ exact-graph accuracy. The full model is exact on every seed by 500 updates. These results refine the short-budget ablation: a sufficiently expressive decoder can recover routed contents without an explicit slot read, so target-to-slot attention is not required for representation. Under this training protocol, however, the explicit read is substantially more parameter-efficient and consistent across seeds. Removing routed writes is more damaging because the source-dependent part of the memory no longer forms separate compartments.

G Multiplicity-Aware Two-Radius Protocol

G.1 Data and batching

Table 2 uses $n = 12$ identifiers, 12 labels, one structural central node, maximum base multiplicity $A = 4$, and replication scales $r \in \{1, 2, 3\}$. For every base example, the labels form a uniformly sampled permutation $\pi \in S_{12}$, and the base multiplicities are sampled independently as $a_i \sim \mathrm{Unif}\{1, 2, 3, 4\}$. The three paired graphs use counts $c_i = ra_i$, so target counts lie in $\{1, \ldots, 12\}$. Source order is independently shuffled in every rendered graph, and target identifiers are placed in a random order.

The directed graph has edges from every source to the central node and from the central node to every target, with mean-normalized aggregation and no reverse edges. This prevents the fixed target set

from entering the source summary through the local path. The largest graph contains 144 sources, one central node, and 12 targets. Smaller graphs are padded to 157 positions; node and source masks remove padding from message passing, pooling, and attention.

Data are generated online. Each training minibatch samples 32 base assignments and renders each at all three scales, giving an effective batch of 96 graphs. Over $10{,}000$ updates, training processes $320{,}000$ base assignments, or $960{,}000$ rendered graphs. Each validation epoch uses 8 fresh minibatches, corresponding to 256 base assignments and 768 rendered graphs.

G.2 Model variants

All variants use 128-dimensional identifier, label, and role embeddings, followed by a two-layer input MLP, LayerNorm, and three residual mean-aggregation MPNN layers. The label and count heads are two-layer MLPs. The count head has 12 classes, where class $c_i - 1$ represents count c_i.

The MPNN baseline has no auxiliary global state. The additive VN mean-pools the sources with a mask, updates one learned virtual state with a three-layer MLP, and broadcasts it to all targets through a residual target MLP with scale 0.2. Both cross-attention variants use $M = 12$ learned static slot addresses and 4 heads. Identifier embeddings are projected into a shared address space used both for source-to-slot writes and target-to-slot reads; dynamic slot content is not reused as an address. The write attends only to source tokens, and both routing directions use temperature 0.35. Slot contents pass through an output projection, LayerNorm, a residual feed-forward block, and a second LayerNorm. The retrieved target context is integrated by FiLM with $\gamma = 0.5 \tanh(\widehat{\gamma})$ and $\beta = \tanh(\widehat{\beta})$, and the final FiLM linear map is initialized with weight standard deviation 10^{-3}.

The anchored variant appends one private, learnable anchor logit to each slot and head before the source softmax. Anchor logits start at zero, so the model initially matches the unit reference mass of $\mathrm{Softmax}_1$ but can learn a different support scale for every slot and head. The per-head log-odds $\log(1 - \alpha_0) - \log \alpha_0$ pass through a two-layer mass MLP and are added to the slot content after content normalization.

G.3 Paired replication protocol

For a fixed base assignment (π, a), the three rendered graphs differ only by the common scale r. Copies of the same identifier have identical identifier, label, role, neighborhood, and address features. Graph cardinality is never given as an input. The source-to-slot attention set contains only source tokens: central and target nodes are excluded because their fixed, non-replicated presence would act as an implicit reference. A normalized source read is therefore exactly invariant across the paired scales, while the correct count labels change.

G.4 Optimization and model selection

All models are trained with AdamW for 200 epochs and 50 minibatches per epoch. The learning rate is 10^{-4} for embeddings, the local backbone, and prediction heads, and 2×10^{-4} for the global block. Weight decay and dropout are zero, and gradients are clipped to norm 5. The loss is $\mathcal{L} = \mathcal{L}_{\mathrm{label}} + w_{\mathrm{count}} \mathcal{L}_{\mathrm{count}}$, with $w_{\mathrm{count}} = 0$ through epoch 30, increased linearly to 0.5 over epochs 31–70, and kept at 0.5 afterwards. At each epoch, metrics are averaged over 8 fresh validation minibatches, and we report the epoch with the best validation BOTH. The result in Table 2 uses seed 0.

G.5 Diagnostics

For one base assignment rendered at all three scales, we compare the full global states with the $r = 1$ state. A normalized memory should show zero maximum and mean difference up to floating-point error, while the anchored state should change with r. We also record write weights, anchor weights, read weights, count accuracy at each scale, and exact graph recovery.

H TREE-NEIGHBORSMATCH PROTOCOL

A complete binary tree of depth r has 2^r candidate leaves. Each candidate carries a label and a distinct local neighbor-count feature; the root carries one such count and must output the corresponding candidate label. As in the controlled benchmark setup, the count is provided directly as an input feature so that the experiment tests communication rather than learning to count. Episodes are generated online. The MPNN baseline uses $r + 1$ local message-passing layers. Global-memory models use the same task generator and are trained for $50{,}000$ updates. The slot models use $M = 64$ memory rows of width 16. Table 3 reports multi-seed estimates at all tested depths. Parameter counts in the main table are those at $r = 5$.

OGB-style virtual node. The OGB-style VN keeps a learned state per graph, initialized at zero. Before every local layer the state is added to every node; after every layer it is updated as $\mathrm{vn} \leftarrow \mathrm{LN}(\mathrm{vn} + \mathrm{MLP}(\mathrm{pool}(h) + \mathrm{vn}))$ with a sum pool, as in the OGB baselines (Hu et al., 2020). Its memory width is the node width. In additional single-seed diagnostic runs at $r = 4$ (width 64, $4{,}000$ updates, 18.8%), doubling the width to 128 (19.0%) or tripling the training length (19.2%) did not materially change accuracy. These checks do not exhaust the tuning space; within the tested range, increasing width or training length alone did not materially improve the result.

Depth $r = 6$. At $r = 6$ (64 candidates), the table reports multi-seed estimates. Slots reach $26.2 \pm 0.2\%$ and anchored slots $25.7 \pm 0.2\%$, compared with 5.1–5.7% for the local and pooled baselines and 1.6% chance. The anchored bilinear realization reaches 100% at $r = 3, 4, 5$ and a mean accuracy of 20.9% at $r = 6$. These results extend the same addressability pattern to a larger candidate set, although the task is no longer solved at this depth.

I STANDARD BENCHMARKS (LRGB)

Protocol. We use Peptides-func (graph classification, macro AP), Peptides-struct (graph regression, MAE) and PascalVOC-SP (node classification, macro F1) from LRGB (Dwivedi et al., 2022), with OGB atom and bond encoders on Peptides and channel-wise feature normalization on PascalVOC-SP. The backbone is GINE with a skip connection around each layer and a multi-layer prediction head, following the retuned protocol of Tönshoff et al. (2023). The epoch is selected on validation and we report the mean test score over five seeds. The final configuration adds one light anchored slot block, without persistence, on the last layer of a GINE that already uses the OGB-style virtual node. We use six layers, backbone width 128, 32 slots of width 32, batch size 128, dropout 0.1 (up to 0.5 for the OGB-VN baseline), and weight decay 0.01.

Table 6: Standard LRGB benchmarks, mean test score over five seeds at the best validation epoch.

Model	Pep-func AP ↑	Pep-struct MAE ↓	VOC-SP F1 ↑
GINE (no virtual node)	0.664	0.261	0.378
GINE + pooled mean VN	0.677	0.252	0.389
GINE + OGB-style VN	0.711	0.239	0.402
GINE + anchored slots	0.701	0.233	0.391
GINE + OGB-style VN + anchored slots	0.712	0.230	0.404
GINE + anchored bilinear memory	0.680	0.247	0.389

Among the tested slot configurations, the light last-layer block gave the most consistent validation results. The anchored bilinear realization gives lower mean scores on these three tasks than the slot/OGB-VN variants, although both multiplicative realizations solve the controlled addressability benchmarks. This table therefore serves as a transfer check rather than a broad performance claim. The addressable cross-attentive ablations were particularly prone to overfitting in our experiments, more tailored optimization could help improve the results.