EXPLORING SYNCHRONY FOR COMMUNICATION IN MODULAR NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

One motivation for modular neural networks is to divide computation across components that learn distinct, reusable functions, which could then be recombined to support compositional generalisation. In practice, modular structure alone does not guarantee that such roles develop, and unrestricted communication between modules can undermine any specialisation that does emerge. This raises the question of how modular communication should be organised. The problem can be viewed as an analogue of the perceptual binding problem, where synchrony has been proposed as a mechanism for coordinating distributed representations. We introduce SyncNet, a modular architecture that separates module content from a dynamical phase state: learnt Kuramoto dynamics evolve the phases, while phase alignment determines communication through a shared channel. We evaluate SyncNet on Sort-of-CLEVR for relational reasoning and on SQOOP for systematic generalisation, where lower pair variety demands more recombination at test time. SyncNet gives the best held-out accuracy of the models compared at the three lowest pair varieties, by 5.1 percentage points at the hardest, and is competitive on Sort-of-CLEVR though below the strongest attention baselines. Test-time overrides show that performance depends on phases that evolve: fixed phases cost as much as no addressing at all, and permuting which module holds which region is nearly as damaging. A content-addressed variant trained from scratch nonetheless matches it, so the benefit appears to come from the modular decomposition rather than synchrony itself.

1 INTRODUCTION

Humans can understand complex problems by breaking them into smaller, reusable concepts and recombining those concepts in new situations. This ability, known as *systematic* or *compositional generalisation*, remains difficult for neural networks to reproduce reliably (Lake & Baroni, 2018; Hupkes et al., 2020; Dziri et al., 2023). Neural networks have become much more capable with scale (Kaplan et al., 2020; Wei et al., 2022), but it remains unclear whether scale alone is enough. This motivates inductive biases that encourage networks to learn more compositional solutions (Goyal & Bengio, 2022; Battaglia et al., 2018).

For a network to recombine its components, those components need stable and reusable functions. This is *functional modularity*: components perform restricted, reusable roles while remaining relatively independent of the rest of the system (Fodor, 1983). It cannot be imposed directly and must emerge through learning, so previous work has used *structural modularity*, dividing a network into separately parameterised components in the hope that specialised functions develop (Schug et al., 2024; Shazeer et al., 2017; Goyal et al., 2021; Andreas et al., 2016). This does not reliably work. Even on tasks with an explicitly modular generative structure, learnt modules do not necessarily align with that structure, and computation can become concentrated in only a few modules (Mittal et al., 2022). How modules communicate also matters: when information can pass freely between sub-networks, increasing connectivity can weaken specialisation (Béna & Goodman, 2025). This leads to our central question: how should a modular system decide which components communicate, and how should information pass between them?

This problem resembles the *binding problem*: how a distributed system represents several entities while preserving which features, roles, or values belong together (Greff et al., 2020). In machine

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.

learning, this is usually studied as the problem of determining which features belong to the same object (Greff et al., 2020; Locatello et al., 2020). Modular systems face a similar problem at the level of components rather than features. We therefore describe this as a *computational binding problem*: structural modularity provides separate components, but does not specify how those components should coordinate. Mechanisms proposed for binding may therefore also provide mechanisms for modular communication.

One possible mechanism is synchrony. In neuroscience, synchrony has been proposed as a way of binding features belonging to the same perceptual object (von der Malsburg, 1981; Singer & Gray, 1995). The *communication through coherence* hypothesis further proposes that the relative phase of neural populations can regulate how effectively they exchange information (Fries, 2005; 2015), although both ideas remain debated (Shadlen & Movshon, 1999; Thiele & Stoner, 2003). In machine learning, phase variables that carry no content themselves have been used to group features into objects (Löwe et al., 2022; 2023), while oscillatory dynamics have also been used directly as part of a model's computation (Miyato et al., 2025; Darlow et al., 2025). In these approaches, however, synchrony acts between individual units. Whether it can instead control communication between separately parameterised modules remains, to our knowledge, unexplored.

We hypothesise that synchrony can provide this coordination and that restricting communication in this way can support compositional generalisation. Structural modularity provides components in which specialised functions may emerge, while synchrony determines how information moves between them. The key question is whether phase synchrony itself is important, or whether a similarly restrictive routing mechanism would work equally well. We make four contributions:

1. We frame modular communication as a computational binding problem and propose synchrony as one possible solution.
2. We introduce SyncNet, which separates module content from phase and uses phase alignment to control communication through a shared channel.
3. We show that SyncNet performs competitive relational reasoning on Sort-of-CLEVR and achieves the highest held-out accuracy among the compared models at the three lowest SQOOP pair-variety settings.
4. Through ablations and overrides, we show that SyncNet relies on its learnt phase dynamics, but that synchrony is not uniquely necessary.

2 Related work

2.1 Modular communication

Modular architectures mainly differ in how they route information between components and combine their outputs (Pfeiffer et al., 2023). Mixture-of-experts models route each input to a sparse subset of experts (Shazeer et al., 2017; Fedus et al., 2022). Neural module networks build computation graphs from modules associated with operations such as attending or comparing. They generalise well to unseen combinations when given the correct layout, but less reliably when that layout must also be learnt (Andreas et al., 2016; Johnson et al., 2017; Bahdanau et al., 2019). Recurrent independent mechanisms use several relatively independent recurrent streams that compete for activation through top-k attention (Goyal et al., 2021). Shared workspace models instead coordinate specialised components through a small set of shared slots, limiting how much information can be broadcast globally (Goyal et al., 2022).

2.2 Synchrony in neural networks

One line of work assigns each unit a phase variable that carries no content itself, separating what the unit represents from the group it belongs to. Early work introduced phase at test time to group features (Reichert & Serre, 2014). Complex-valued autoencoders made this grouping trainable, with magnitude representing feature presence and relative phase representing group membership (Löwe et al., 2022). Rotating Features extended scalar phase to higher-dimensional orientation vectors (Löwe et al., 2023). SyncNet uses the same separation between content and phase, but applies it

to modules rather than individual features. It also uses phase alignment to control communication, rather than simply to identify which features belong together.

A second line of work uses oscillatory dynamics as part of the computation itself. Artificial Kuramoto Oscillatory Neurons replace scalar activations with unit vectors that evolve according to a natural frequency, coupling between oscillators, and an input-dependent stimulus, improving unsupervised object discovery and reasoning (Miyato et al., 2025). Winfree-oscillator networks take a similar approach using structured Winfree dynamics (Dai & Song, 2026). SyncNet uses the spherical Kuramoto update of Miyato et al. (2025) in its perceptual front end. However, in both of these models the oscillators are individual neurons, so synchrony does not control communication between separately parameterised modules. The Continuous Thought Machine also uses synchrony, but in a different way: computation unfolds over internal steps and synchrony is measured as pairwise correlation between neuron activation histories. This synchrony forms part of the model's representation rather than controlling communication between modules (Darlow et al., 2025).

3 METHOD

SyncNet is a modular architecture in which synchrony has two roles. First, it groups the spatial feature map into modules. More importantly, it controls communication between those modules through their relative phases. The design follows two main principles. Each module maintains a content state containing its current information and a phase describing its relationship to other modules. This separates what a module represents from which modules it communicates with. All modules then write to and read from a single shared channel, with the strength of communication determined by phase alignment.

Computation occurs in four main stages. An encoder turns the image and question into a feature map. The feature map is then split into spatial cells, and those cells are assigned across M modules. Each module holds a content state and a phase state. The modules then exchange information over T internal steps, while the content and phase states evolve. Finally, a readout maps the state of a separate head module to the output.

$$F = \mathcal{E}(I, \bar{q}) \qquad \text{encode} \tag{1}$$

$$\{X_k, h_k^{(0)}, z_k^{(0)}\}_{k=1}^{M} = \mathcal{B}(F) \qquad \text{bind} \tag{2}$$

$$(h^{(t)}, z^{(t)}) = \mathcal{S}(h^{(t-1)}, z^{(t-1)}, X), \quad t = 1, \dots, T \qquad \text{communicate} \tag{3}$$

$$y = \mathcal{R}(h_0^{(T)}, \bar{q}) \qquad \text{read.} \tag{4}$$

Here, X_k is the part of the feature map assigned to module k, $h_k^{(t)}$ its content state at internal step t, and $z_k^{(t)}$ its phase. Throughout, M counts the perceptual modules, indexed $k = 1, \dots, M$; the head module introduced below is indexed $k = 0$, so the $M + 1$ participants in the shared channel are indexed $k = 0, 1, \dots, M$. The remainder of this section defines each component following the numbered stages in Figure 1.

Spherical phase. Phase is represented throughout as a unit vector $z_i \in S^{d-1}$ rather than as a point on a circle. Following AKOrN, it evolves using a fixed number of explicit Euler steps of the spherical Kuramoto dynamics (Miyato et al., 2025; Kuramoto, 1975):

$$z_i \leftarrow \Pi\left(z_i + \gamma\left(\Omega_i z_i + \mathrm{Proj}_{z_i}\left(\sum_j J_{ij} z_j + c_i\right)\right)\right), \tag{5}$$

where Ω_i is an antisymmetric matrix controlling the oscillator's natural rotation, which keeps its contribution tangent to the sphere, J_{ij} is the coupling between oscillators, c_i is a conditional stimulus, and γ is the step size. Ω_i and J_{ij} are learnt in both the field and the module dynamics, with Ω_i antisymmetrised by construction; γ is a fixed hyperparameter, with separate values for the field and the modules given in Appendix C. The tangent-space projection is $\mathrm{Proj}_z(v) = v - \langle v, z\rangle z$, and $\Pi(v) = v/\|v\|$ renormalises the result to unit length. Positive coupling pulls oscillators together, while negative coupling pushes them apart. At $d_x = 2$ this reduces to the classical scalar Kuramoto model (Miyato et al., 2025).

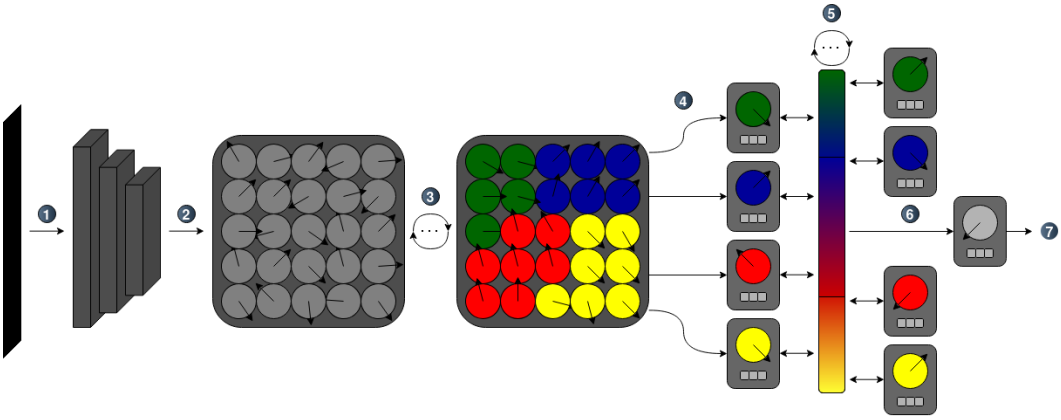


Figure 1: Overview of SyncNet, shown with four modules for clarity. (1) A convolutional encoder encodes the image into a spatial feature map. (2–3) Oscillators attached to the spatial features evolve under local coupling and feature-conditioned stimuli, allowing related cells to align in phase. Colour denotes phase group throughout the figure. (4) The resulting phase groups initialise the module content states and phases. (5) Modules communicate over a shared phase-addressed channel while their phases evolve over internal steps. (6) A head module participates in the same communication process but does not observe the image directly. (7) The final answer is read from the head state.

(1) The encoder. A convolutional encoder maps the image I, conditioned on the encoded question $\bar{q}$, to a feature map F over an $S \times S$ spatial grid, where F_p is the feature at cell p. The encoder depends on the task but is shared with the baselines on that task, so differences between models do not come from different feature extractors.

(2–3) The oscillator field. Each spatial cell p carries K spherical oscillators of dimension d_x, written together as x_p. These are initialised from the local feature using a linear projection, $x_p^{(0)} = \Pi(W_x F_p)$ applied to each oscillator in turn, and then evolved for T_x steps of Equation 5 at step size γ_x. The oscillator dimension d_x is a property of the field and is distinct from the module phase dimension d of the module dynamics below; values for both are given in Appendix C. The update is applied across the spatial grid with learnt convolutional coupling between neighbouring cells and a feature-conditioned stimulus $C = W_c F$. This follows the oscillator layer of AKOrN (Miyato et al., 2025); its settings are given in Appendix C. Together, the local coupling and feature-conditioned stimulus allow related cells to move towards useful phase groupings.

(4) Module initialisation. Each module k has a learnt starting phase $\bar{a}_k$, with a small perturbation so that its initial phase is not identical on every pass:

$$a_k^{(0)} = \Pi(\bar{a}_k + \sigma_k \epsilon_k), \qquad \epsilon_k \sim \mathcal{N}(0, I). \tag{6}$$

Distinct anchors lead modules to claim different cells and so to hold different information, which is what gives them something to exchange, and learning them makes that starting point repeatable across passes. Modules then compete for spatial cells according to phase alignment over T_b iterations:

$$\alpha_{kp}^{(t)} = \underset{k}{\mathrm{softmax}}\left(\lambda \langle a_k^{(t-1)}, x_p \rangle + b_k(p)\right), \qquad a_k^{(t)} = \Pi\left(\sum_p \alpha_{kp}^{(t)} x_p\right), \tag{7}$$

where λ controls the sharpness of the competition and is learnt from a fixed initial value, and $b_k(p)$ is a learnt spatial preference. The anchors occupy the same space as the field, so a_k is likewise a set of K unit vectors of dimension d_x, and $\langle a_k, x_p \rangle$ denotes the mean of the K corresponding oscillator inner products. After the final iteration, each module receives the weighted average of its assigned features,

$$X_k = \frac{\sum_p \alpha_{kp}^{(T_b)} F_p}{\sum_p \alpha_{kp}^{(T_b)}}, \tag{8}$$

with initial channel phase $z_k^{(0)} = \Pi(W_a a_k^{(T_b)})$, where W_a maps the Kd_x anchor coordinates into the d-dimensional phase space and initial content state $h_k^{(0)} = \mathcal{I}(X_k, \bar{q})$, where $\mathcal{I}$ is a two-layer multi-layer perceptron (MLP) applied to the concatenation $[X_k; \bar{q}]$ and is shared across modules.

(5) The module update. Module k maintains a content state $h_k^{(t)} \in \mathbb{R}^{d_h}$ and a phase $z_k^{(t)} \in S^{d-1}$. Each of the T internal steps consists of communication, a content update, and a phase update:

$$r_k^{(t)} = \mathcal{C}(m^{(t)}, z^{(t-1)})_k, \qquad m_k^{(t)} = W_m h_k^{(t-1)} \qquad \text{communicate} \tag{9}$$

$$h_k^{(t)} = \mathrm{GRU}_k\left([X_k; r_k^{(t)}], h_k^{(t-1)}\right) \qquad \text{update content} \tag{10}$$

$$z_k^{(t)} = \Phi(z^{(t-1)}, h^{(t)})_k \qquad \text{update phase,} \tag{11}$$

where $m_k^{(t)}$ is module k's message and $r_k^{(t)}$ is the information it receives from the shared channel $\mathcal{C}$. Each module has its own gated recurrent unit (GRU) cell GRU_k, allowing different modules to learn different computations. These separate recurrent parameters provide SyncNet's structural modularity. The message projection W_m, channel, and phase dynamics are shared across modules.

The shared channel. Module j writes its message in the direction of its phase, $B = \sum_j m_j z_j^\top \in \mathbb{R}^{d_m \times d}$, and module k reads back the component aligned with its own phase:

$$\mathcal{C}(m, z)_k = \frac{1}{M}\left(Bz_k - m_k\right) = \frac{1}{M}\sum_{j \neq k} m_j \langle z_j, z_k \rangle, \tag{12}$$

so a sender with the same phase contributes fully, a sender in antiphase contributes negatively, and an orthogonal sender does not contribute. Subtracting m_k removes the receiver's own message. All indices run over $j \in \{0, 1, \ldots, M\}$: the M perceptual modules together with the head module described below, which writes to and reads from the channel on the same terms. Each receiver therefore averages over the M other participants, which is the normalising constant. This follows the communication-through-coherence hypothesis, where relative phase controls communication strength (Fries, 2005).

Messages from modules with similar phases therefore combine, while messages in different phase directions remain separated. The phase dimension d limits how many distinct directions can coexist. The channel is intentionally lossy, since each module receives only information aligned with its current phase, and unrestricted communication can weaken specialisation (Béna & Goodman, 2025). Equation 12 can also be viewed as a form of linear attention (Katharopoulos et al., 2020; Schlag et al., 2021), with the same phase acting as both query and key, so alignment is symmetric and phases persist as dynamical states rather than being recomputed from the current content.

The phase dynamics. Module phases evolve using Equation 5, with the coupling between modules modulated by their current content states:

$$\Phi(z, h)_k = \Pi\left(z_k + \gamma\left(\Omega_k z_k + \mathrm{Proj}_{z_k}\left(\sum_j J_{kj}\kappa_{kj}(h) z_j + c_k\right)\right)\right), \tag{13}$$

$$\kappa_{kj}(h) = \mathrm{MLP}_\kappa\big([h_k; h_j]\big) \in (-1, 1), \qquad c_k = W_c h_k. \tag{14}$$

The effective coupling $J_{kj}\kappa_{kj}(h)$ is a learnt base coupling scaled by a small network that reads the two content states. Positive values pull the phases together, while negative values push them apart. The states are given in receiver–sender order, so the coupling is asymmetric, $\kappa_{kj} \neq \kappa_{jk}$. The stimulus c_k also allows a module's own content to directly influence its phase. The output of MLP_κ passes through a $\tanh$, which gives the stated range.

Content therefore affects communication only indirectly. As the module states change, the couplings change, which changes the phase relationships and so the strength of communication at the next step.

(6–7) The head and the readout. The head is an additional module, indexed by $k = 0$, with its own recurrent cell. It writes to and reads from the shared channel through its phase and updates that

phase using the same dynamics as the other modules. However, it has no direct access to the image. Its content state is initialised from the encoded question, $h_0^{(0)} = \mathcal{I}_0(\bar{q})$, its phase from a random unit vector, and its module input is fixed to $X_0 = 0$ because it receives no spatial cells. After T steps, the prediction is read from its final state, $y = \mathrm{MLP}_y([h_0^{(T)}; \bar{q}])$, which is the readout $\mathcal{R}$ of Equation 4. Image information can therefore affect the prediction only after being assigned to a perceptual module and communicated to the head through the shared channel.

4 Experimental setup

4.1 Datasets

Both tasks are diagnostic probes rather than performance benchmarks, chosen because each isolates one capability under controlled conditions. Sort-of-CLEVR (Santoro et al., 2017) supplies questions as fixed binary vectors over synthetic scenes, removing language and keeping perception simple so that differences between models reflect relational computation rather than representation (Marza et al., 2022). Its three question families, non-relational, binary and ternary, give graded relational complexity as more objects must be considered jointly; we include the ternary family of Kim (2018). It does not test systematic generalisation, since its questions are fixed codes that do not compose and its standard split holds out no novel combinations.

SQOOP (Bahdanau et al., 2019) supplies that test. Each question names two of 36 objects and one of four spatial relations, and asks whether the relation holds. Difficulty is controlled by the *pair variety*: the number of distinct right-hand objects paired with each left-hand object during training, with the remaining pairings held out for evaluation. Lower pair variety exposes the model to fewer combinations and demands more recombination, so the axis directly parameterises how much systematic generalisation is required. This control is why a larger benchmark would not substitute: no standard dataset exposes a comparable knob, and without it the quantity we report cannot be measured. The cost is that SQOOP tests only simple spatial relations, so strong held-out accuracy is evidence about recombination rather than about compositional reasoning in general. Example scenes and questions from both tasks are shown in Figure 3.

4.2 Baselines and protocol

All models on a task share the same convolutional image encoder where possible, so that differences reflect the reasoning and communication mechanism rather than feature extraction: the encoder of Santoro et al. (2017) on Sort-of-CLEVR and of Bahdanau et al. (2019) on SQOOP, with a patch embedding for the token-based models. Questions are encoded by an MLP on Sort-of-CLEVR and a long short-term memory network (LSTM) on SQOOP.

We compare against six baselines, defined in full in Appendix B: a **question-only** model that fixes the floor, the **convolutional** baselines CNN+MLP and Conv+LSTM (Santoro et al., 2017; Bahdanau et al., 2019), the **Relation Network** (Santoro et al., 2017), **FiLM** (Perez et al., 2018), a **Transformer** (Vaswani et al., 2017), and the **Shared Workspace** (Goyal et al., 2022). The last is the closest comparison, restricting communication to a small set of persistent slots as SyncNet restricts it to one channel, but selecting by content rather than phase.

We do not report an end-to-end AKOrN baseline because AKOrN was not designed or evaluated for question-conditioned tasks (Miyato et al., 2025). Constructing such a baseline would therefore require introducing our own conditioning mechanism, making it difficult to distinguish the behaviour of AKOrN itself from that of our adaptation. We also do not report a neural module network. Bahdanau et al. (2019) show on SQOOP that one generalises well there only when both the layout and the module parametrisation are supplied by hand, and that inducing either end-to-end fails.

Optimisation settings are held fixed across models and conditions, with all deviations recorded in Appendix D; the only exceptions are the two baselines that failed to train reliably under the shared protocol and were tuned until they did. Every model is run for five seeds unless stated otherwise, and results are reported as mean $\pm$ standard deviation. For each run we select the checkpoint with the highest validation accuracy and evaluate it once on the test set. On SQOOP the validation set contains only pairings seen during training, so held-out pairings are never used for model selection.

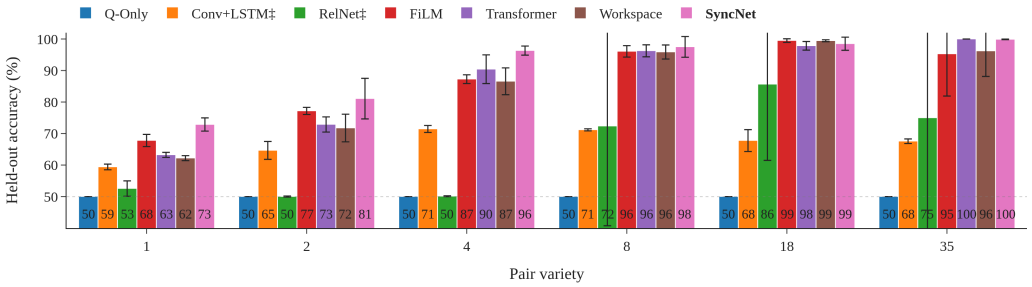


Figure 2: SQOOP held-out accuracy (mean ± s.d. over five seeds) across pair-variety settings. The Relation Network and Conv+LSTM, marked ‡, use their tuned configurations. At pair variety 35 the held-out split matches the validation distribution.

Table 1: Sort-of-CLEVR: test accuracy in per cent, overall and by question family (mean over five seeds), with parameter counts in millions. Higher accuracy is better (↑); fewer parameters is better (↓). Our method is shown in bold. Per-subtype results are given in Appendix E.

	SyncNet	*Q-Only*	CNN+MLP	RelNet	FiLM	Trans.	Works.
Params (M) ↓	1.03	0.02	6.11	0.44	2.41	0.63	0.58
Overall Acc. (%) ↑	91.9	49.4	66.3	91.3	95.5	93.7	94.7
Non-relational Acc. (%) ↑	99.9	50.7	71.7	99.9	100.0	100.0	100.0
Binary Acc. (%) ↑	97.0	43.6	73.0	97.2	98.9	98.3	98.8
Ternary Acc. (%) ↑	78.9	54.1	54.3	76.7	87.6	82.9	85.4

5 Results

5.1 Compositional generalisation: SQOOP

Every model reaches near-perfect validation accuracy at every setting, so the comparison of interest is held-out accuracy, shown in Figure 2. At pair varieties 8, 18 and 35, FiLM, the Transformer, the Shared Workspace and SyncNet perform similarly: more pairings have been seen during training, so little recombination is required. The differences open as pair variety falls. SyncNet reaches 72.9%, 81.1% and 96.3% accuracy at pair varieties 1, 2 and 4, which is 5.1, 3.9 and 5.9 percentage points above the next best model at each setting. Its advantage is therefore largest exactly where the fewest combinations have been seen. The Conv+LSTM and Relation Network baselines did not train reliably under the shared protocol and are reported in their tuned configurations; Appendix D records the failure mode and the adjustments made. SyncNet gives the strongest out-of-distribution performance overall, with its clearest margin on the splits demanding the most recombination, which suggests the architecture supplies a useful inductive bias for systematic generalisation. However, these results alone do not establish that synchrony is responsible.

5.2 Relational reasoning: Sort-of-CLEVR

Every model except CNN+MLP and the question-only baseline is near ceiling on the non-relational questions, so Table 1 turns on the binary and ternary families. On binary questions SyncNet reaches 97.0% accuracy, the lowest of the five relational models but within two percentage points of FiLM at 98.9%. Overall it reaches 91.9%, above the Relation Network at 91.3% and below the remaining three by at most 3.6 percentage points, at 1.03M parameters, more than the Transformer and Shared Workspace but under half of FiLM, so size does not account for the ordering.

The gap widens on ternary questions, where SyncNet reaches 78.9% accuracy against 82.9%, 85.4% and 87.6% for the Transformer, Shared Workspace, and FiLM. SyncNet is therefore a capable relational reasoner but not the strongest, and the cost of restricting communication to a single shared

Table 2: Component ablations. Each row reports test accuracy in per cent (mean ± s.d. over five seeds) for the main model with one component changed; higher is better (↑). Our method is shown in bold. SQOOP reports held-out accuracy at pair variety 1, where 50.0 is chance.

	Sort-of-CLEVR				SQOOP
Ablation	**Overall** ↑	**Non-rel.** ↑	**Binary** ↑	**Ternary** ↑	**Held-out** (1) ↑
SYNCNET (main)	91.9 ± 1.0	99.9	97.0	78.9	72.9 ± 2.1
no spatial preference	90.9 ± 3.6	99.9	96.3	76.5	50.0 ± 0.1
no structural modularity	90.7 ± 0.8	99.9	96.6	75.6	67.0 ± 4.6
phase + softmax	91.6 ± 0.6	99.9	97.3	77.6	69.3 ± 3.7
content + linear	90.7 ± 0.8	99.9	96.8	75.5	70.2 ± 1.5
content + softmax	91.1 ± 0.8	99.9	97.2	76.2	72.5 ± 4.4

channel grows as information from more objects must be combined at once. This is the boundary of the mechanism: it does not confer a general reasoning advantage.

5.3 COMPONENT ABLATIONS

Each variant replaces exactly one component of the main model and is trained from scratch under the shared protocol, evaluated on Sort-of-CLEVR and on SQOOP at pair variety 1; Appendix F defines them. *No spatial preference* sets $b_k(p) = 0$, and *no structural modularity* replaces the per-module recurrent cells with a single shared cell. The three communication variants complete the combinations of address source, phase or content, with combination rule, linear or softmax. *Content + softmax* also replaces the shared channel with direct attention between modules, so it moves the address source and the communication topology together and isolates neither; *content + linear* is the controlled comparison for the address source.

Changes with little effect on Sort-of-CLEVR often have a much larger effect on SQOOP. Removing the spatial preference costs one percentage point on Sort-of-CLEVR but raises seed variation from ± 1.0 to ± 3.6, and on SQOOP drops held-out accuracy from 72.9% to chance. The failure matches the optimisation problem seen in the two unstable baselines, with gradient norms falling early and runs not recovering. On SQOOP no seed escapes chance, at 50.0 ± 0.1, so the failure is complete rather than intermittent, while on Sort-of-CLEVR the same ablation is survivable but less reliable. The spatial preference therefore governs whether a useful decomposition can be learnt at all, rather than the capacity of that decomposition once learnt. Removing structural modularity shows the same pattern more weakly, at 90.7% and 67.0%. Collapsing M recurrent cells into one also reduces the parameter count of that component, and the shared cell is not widened to compensate, so part of this drop may reflect capacity rather than the loss of specialisation.

The communication variants behave differently. They give 91.6% and 69.3% for *phase + softmax*, 90.7% and 70.2% for *content + linear*, and 91.1% and 72.5% for *content + softmax*, against 91.9% and 72.9% for the main model. All four are within noise of one another at five seeds, and there is no consistent ordering, since changing both the address source and the combination rule outperforms changing either alone. The address source is not what separates these models.

5.4 TEST-TIME OVERRIDES

We apply four overrides to the trained model without retraining, so that a drop in accuracy shows what the model relies on. Each acts on the perceptual modules only, and the head's phase evolves as trained. The head's phase starts from a random vector and carries no information about the scene, so freezing it too would disable the readout and confound the result.

Freeze holds each module's phase at its post-binding value, removing evolution but keeping the initial assignment. *Zero* sets every module phase to one common vector, removing addressing but leaving the messages unchanged. The two shuffle variants permute across modules after binding. *Shuffle-phase* permutes the phases alone, so each module keeps its assigned content. *Shuffle-binding*

Table 3: Test-time overrides applied to the trained main model. Values are the change in test accuracy in percentage points relative to the unmodified model (mean $\pm$ s.d. over five seeds); a more negative value means a more damaging override.

Task	**Freeze**	**Zero**	**Shuffle-phase**	**Shuffle-binding**
Sort-of-CLEVR	-42.7 $\pm$ 25.4	-41.9 $\pm$ 25.9	-5.6 $\pm$ 10.0	-31.9 $\pm$ 3.7
SQOOP (1)	-22.8 $\pm$ 2.7	-24.2 $\pm$ 3.6	-1.9 $\pm$ 1.8	-23.2 $\pm$ 2.4

permutes phase and content together, so module k receives module $\pi(k)$'s binding intact. Appendix F defines all four.

Freeze and Zero cost much the same, 42.7 and 41.9 percentage points of accuracy on Sort-of-CLEVR and 22.8 and 24.2 on SQOOP. Both leave SQOOP at chance and Sort-of-CLEVR at the question-only floor of 49.4%. Phase addressing is therefore load-bearing, but fixed phases are worth no more than no addressing at all. What the model uses is how the phases evolve, not how they differ at the first step.

Permuting the phases alone costs 5.6 percentage points on Sort-of-CLEVR and 1.9 on SQOOP. Each module keeps its content, and the set of phases and the angles between them are unchanged, so only which module holds which phase differs. The model is close to indifferent: it depends on how phases relate as they evolve, not on which phase a module starts from.

Permuting phase and content together costs 31.9 percentage points and 23.2, again leaving SQOOP at chance. The pairing of phase to content survives, so what changes is which module holds which region. The modules are therefore not interchangeable, since the private cells and the readout depend on where a region's binding arrives. This is our most direct evidence that the modules specialise, and it agrees with the shared-cell ablation above.

Taken with the ablations, this locates the mechanism. Synchrony is load-bearing once learnt, and removing phase evolution from a trained model drops it to the floor. It is not uniquely necessary, since a content-addressed model trained from scratch reaches comparable accuracy. What the two share is the modular decomposition, and disturbing that decomposition costs as much as removing synchrony. The evidence supports the stability of the decomposition rather than the mechanism that addresses it.

Appendix E reports sweeps over the phase dimension, the number of internal steps and the number of modules. Accuracy rises with d on Sort-of-CLEVR and flattens once three-way interactions can be separated, SQOOP benefits from more internal steps, and the module count has little effect up to $M = 8$.

6 Conclusion

Synchrony can coordinate communication between the components of a modular network. SyncNet performs non-trivial relational reasoning while restricting all inter-module traffic to a single phase-addressed channel, and reaches the highest held-out accuracy among the models compared at the three lowest pair-variety settings, with its clearest margin where the fewest combinations have been seen during training. Overrides confirm the trained model depends on phases that evolve: holding them fixed costs as much as removing addressing altogether, and permuting which module holds which region costs almost as much again. Ablations show that a content-addressed variant learns an equally good solution from scratch, and that a stable modular decomposition matters more than the mechanism used to address it.

Three limits bound these claims. The evidence comes from two diagnostic tasks chosen for the control they afford, and we have no evidence about behaviour at scale. Functional modularity is only partly measured. The overrides show that the modules are not interchangeable, since permuting which module holds which region costs a trained model most of its accuracy, but they say nothing about what any module computes, so whether the roles acquired are reusable across tasks is untested and is the extension we would pursue first. And while synchrony is sufficient and load-bearing once learnt, these experiments do not show it is preferable to more general routing.

AI USE STATEMENT

We used generative AI assistants during this project, and disclose their use as follows. They were used to format and debug LaTeX tables and figures, with every reported number checked against its source; to assist with sentence-level wording and to identify repetition, with all changes reviewed for consistency with the intended content; to explain technical concepts during the research process, with every claim in this paper verified against the primary source; to help read other research codebases when adapting baseline models, so that those adaptations remained faithful to the original implementations; and to help restructure and simplify existing code for readability without changing its behaviour. Generative AI can be fluent and wrong at the same time, so all AI-generated suggestions were independently reviewed before use. The framing of the problem, the research questions, the methodology, the experimental design, the implementation decisions, and the analysis, interpretation and conclusions presented here are the authors' own. Specifically, ChatGPT (GPT-5.6 Sol) was used in relation to the above.

ETHICS STATEMENT

This work raises no ethical concerns. It involves no human participants and no personal data, and both datasets are generated synthetically, so no questions of consent or privacy arise. The datasets contain geometric shapes and alphanumeric glyphs, so the representation concerns attaching to natural image corpora do not apply. The models are small relational reasoners trained on synthetic scenes and have no application we are aware of that carries a foreseeable risk of harm. The authors declare no conflicts of interest and no sponsorship that could have influenced the work. All experiments were run on a single GPU, with individual training runs averaging roughly 45 minutes, so the compute cost of the work was modest.

REPRODUCIBILITY STATEMENT

The architecture is specified in full in Section 3, with the oscillator field's settings given in Appendix C. Dataset generator settings for both tasks, including every pair-variety setting, are in Appendix A; both datasets are synthetic and can be regenerated deterministically. Baseline architectures are in Appendix B, per-model configurations in Appendix C, and the shared optimisation protocol in Appendix D, which also records every deviation from it by model and task. The ablation variants and the test-time overrides are defined exactly in Appendix F. Every reported number is a mean over five seeds with the standard deviation given where applicable, and checkpoint selection is described in Appendix D. The source code is included as a zipped archive in the supplementary material.

REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations (ICLR)*, 2019.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Gabriel Béna and Dan F. M. Goodman. Dynamics of specialization in neural modules under resource constraints. *Nature Communications*, 16(1):187, 2025. doi: 10.1038/s41467-024-55188-9.

Jiawen Dai and Yue Song. Winfree oscillatory neural network. *arXiv preprint arXiv:2605.20922*, 2026.

Luke Darlow, Ciaran Regan, Sebastian Risi, Jeffrey Seely, and Llion Jones. Continuous thought machines. *arXiv preprint arXiv:2505.05522*, 2025.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.18654.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23:1–40, 2022.

Jerry A. Fodor. *The Modularity of Mind: An Essay on Faculty Psychology*. MIT Press, 1983.

Pascal Fries. A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends in Cognitive Sciences*, 9(10):474–480, 2005. doi: 10.1016/j.tics.2005.08.011.

Pascal Fries. Rhythms for cognition: communication through coherence. *Neuron*, 88(1):220–235, 2015. doi: 10.1016/j.neuron.2015.09.034.

Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 478(2266):20210068, 2022.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:1909.10893.

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace. In *International Conference on Learning Representations (ICLR)*, 2022.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research (JAIR)*, 67:757–795, 2020.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 5156–5165, 2020. arXiv:2006.16236.

Heecheol Kim. relational-networks: Pytorch implementation of relation networks. `https://github.com/kimhc6028/relational-networks`, 2018. Accessed August 2026.

Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In Huzihiro Araki (ed.), *International Symposium on Mathematical Problems in Theoretical Physics*, volume 39 of *Lecture Notes in Physics*, pp. 420–422. Springer, 1975.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, 2018. arXiv:1711.00350.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2006.15055.

Sindy Löwe, Phillip Lippe, Maja Rudolph, and Max Welling. Complex-valued autoencoders for object discovery. *Transactions on Machine Learning Research (TMLR)*, 2022. arXiv:2204.02075.

Sindy Löwe, Phillip Lippe, Francesco Locatello, and Max Welling. Rotating features for object discovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.00600.

Pierre Marza, Corentin Kervadec, Grigory Antipov, Moez Baccouche, and Christian Wolf. An experimental study of the vision-bottleneck in vqa. *arXiv preprint arXiv:2202.06858*, 2022.

Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2206.02713.

Takeru Miyato, Sindy Löwe, Andreas Geiger, and Max Welling. Artificial Kuramoto oscillatory neurons. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.13821.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. arXiv:1709.07871.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *Transactions on Machine Learning Research (TMLR)*, 2023. arXiv:2302.11529.

David P. Reichert and Thomas Serre. Neuronal synchrony in complex-valued deep networks. In *International Conference on Learning Representations (ICLR)*, 2014. arXiv:1312.6115.

Adam Santoro, David Raposo, David G. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 9355–9366, 2021. arXiv:2102.11174.

Simon Schug, Seijin Kobayashi, Yassir Akram, Maciej Wołczyk, Alexandra Proca, Johannes von Oswald, Razvan Pascanu, João Sacramento, and Angelika Steger. Discovering modular solutions that generalize compositionally. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2312.15001.

Michael N. Shadlen and J. Anthony Movshon. Synchrony unbound: A critical evaluation of the temporal binding hypothesis. *Neuron*, 24(1):67–77, 1999.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1701.06538.

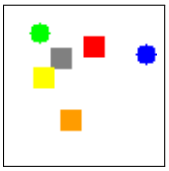
Wolf Singer and Charles M. Gray. Visual feature integration and the temporal correlation hypothesis. *Annual Review of Neuroscience*, 18:555–586, 1995.

Alexander Thiele and Gene Stoner. Neuronal synchrony does not correlate with motion coherence in cortical area MT. *Nature*, 421:366–370, 2003.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017. arXiv:1706.03762.

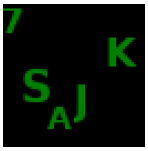
Christoph von der Malsburg. The correlation theory of brain function. Technical Report Internal Report 81-2, Max-Planck-Institute for Biophysical Chemistry, 1981. Reprinted in Models of Neural Networks II, 1994.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research (TMLR)*, 2022. arXiv:2206.07682.



family	question
Non-rel.	Shape of the blue object? **circle**
Binary	Shape of the object closest to the green object? **rectangle**
Ternary	How many objects lie in the box between the blue and yellow objects? **1**

(a) Sort-of-CLEVR



question
7 above S? **yes**
7 left of S? **no**

(b) SQOOP

Figure 3: Example scenes and questions from the two tasks. Generator settings are given in Table 4.

A Dataset configurations

Figure 3 shows example scenes and questions from both tasks, and Table 4 gives the generator settings used to produce them. Separate datasets are generated for each pair-variety setting of SQOOP.

Setting	Sort-of-CLEVR	SQOOP
Training examples	1,080,000 (36,000 scenes)	1,080,000
Evaluation examples per split	30,000 (1,000 scenes)	25,600
Image size (px)	75×75	64×64
Objects per scene	6	5
Object size (px)	5	10 – 15
Questions per scene	30 (10 per family)	—
Object vocabulary	—	36
Spatial relations	—	4
Pair variety	—	[1, 2, 4, 8, 18, 35]

Table 4: Dataset Generator configuration for Sort-of-CLEVR and SQOOP, separate datasets are generated for each of the listed pair-variety settings.

B Baseline architectures

Notation follows Section 3: I is the image, q the question, $\bar{q}$ its encoding, $F = \mathrm{enc}(I)$ the image feature map, F_p the feature at spatial location p, and ξ_p its coordinates. Components shared across models are listed in Table 5 of Appendix C, which also gives the per-model configurations; deviations from the shared optimisation protocol are recorded in Appendix D.

Question-only. The question-only baseline predicts the answer from the encoded question without using the image. It shows how much performance can be obtained from the question and answer distributions alone. This is particularly useful on Sort-of-CLEVR, where answer priors differ across question types.

Convolutional baselines. The convolutional baselines test how much of each task can be solved without an explicit relational or communication mechanism. The image feature map is flattened, concatenated with the encoded question, and passed through an MLP:

$$\hat{y} = \mathrm{MLP}\left([\mathrm{vec}(F); \bar{q}]\right). \tag{15}$$

On Sort-of-CLEVR, the question is encoded with an MLP. On SQOOP, it is encoded with an LSTM. These give the CNN+MLP and Conv+LSTM baselines used in the original papers (Santoro et al., 2017; Bahdanau et al., 2019).

Relation Network. The Relation Network explicitly models pairwise relations between spatial features (Santoro et al., 2017). Each ordered pair of features is processed by the same relation

function, conditioned on the question:

$$\hat{y} = f_\phi \left(\sum_p \sum_{p'} g_\theta \left([o_p; o_{p'}; \bar{q}] \right) \right), \qquad o_p = [F_p; \xi_p]. \tag{16}$$

Here, g_θ represents the relation between each pair of spatial features, while f_ϕ maps the combined relations to the final answer. Unlike SyncNet, relational interactions are built directly into the architecture.

FiLM. FiLM uses the question to modulate intermediate visual features (Perez et al., 2018). Writing $H^{(0)} = F$, the question representation produces feature-wise scaling and shifting parameters $(\gamma^{(b)}, \beta^{(b)})$ for each residual block:

$$u^{(b)} = \mathrm{ReLU}\left(W_1^{(b)} * [H^{(b-1)}; \xi] \right), \tag{17}$$

$$H^{(b)} = \mathrm{ReLU}\left((1 + \gamma^{(b)}) \odot \mathrm{BN}\left(W_2^{(b)} * u^{(b)} \right) + \beta^{(b)} \right) + u^{(b)}. \tag{18}$$

Here, $\gamma^{(b)}$ and $\beta^{(b)}$ follow the notation of Perez et al. (2018) and are unrelated to the integration step size and the competition sharpness of Section 3. The question can therefore influence visual processing throughout the network. Unlike the Relation Network, FiLM does not explicitly model pairwise relations.

Transformer. The Transformer provides a general-purpose baseline based on self-attention (Vaswani et al., 2017). The image is represented as a sequence of spatial tokens with positional information and passed through a standard Transformer encoder.

On Sort-of-CLEVR, the encoded question is added as another token. On SQOOP, the question representation is concatenated to each image token. In both cases, the final answer is predicted from a classification token.

Unlike SyncNet, the Transformer determines interactions directly from token content through self-attention. It does not maintain a separate state controlling communication.

Shared Workspace. The Shared Workspace restricts communication through a small set of persistent workspace slots (Goyal et al., 2022). Our implementation is based on the Transformer baseline and uses the same tokenisation, positional information, and question conditioning. However, direct self-attention between tokens is replaced by communication through the workspace.

During the write step, the workspace slots attend over the input tokens. Only a fixed fraction of tokens with the highest attention weights contribute to each slot. During the broadcast step, the tokens attend over the updated workspace:

$$W^{(\ell+1)} = \mathrm{Update}\left(W^{(\ell)}, \mathrm{MHA}_{\text{top-}k}\left(W^{(\ell)}, Z^{(\ell)} \right) \right), \tag{19}$$

$$Z^{(\ell+1)} = \mathrm{Block}\left(Z^{(\ell)} + \mathrm{MHA}\left(Z^{(\ell)}, W^{(\ell+1)} \right) \right). \tag{20}$$

The same layer is applied recurrently with shared weights, following the recurrent form of the original architecture. This is the closest baseline to SyncNet because both restrict communication through a shared channel. The main difference is that the Shared Workspace controls access using content-based attention, while SyncNet uses phase.

C Model configurations

The oscillator field applies Equation 5 across the $S \times S$ spatial grid. Each cell carries K oscillators of dimension d_x, so the field has Kd_x channels at each spatial position and each oscillator is renormalised to unit length independently. The coupling J is a learnt convolution over the grid, so a cell is coupled only to its neighbourhood and the same coupling applies at every position, and the stimulus is $C = W_c F$, a pointwise linear map of the feature map, which ties each oscillator's preferred direction to the feature at its own cell. Both Ω and J are learnt, as in Section 3.

Component	Sort-of-CLEVR	SQOOP
Shared encoder	Four 3×3 convolutions, stride 2, 128 channels, batch norm and ReLU; 5×5 output map	Six 3×3 convolutions, two max-pools, 64 channels, batch norm and ReLU; 16×16 output map
Patch embedding	Single 5×5 convolution, stride 5; 15×15 tokens	Single 4×4 convolution, stride 4; 16×16 tokens
Question encoder	Identity or 2-layer MLP for CNN+MLP	LSTM

Table 5: Components shared across models.

Setting	Sort-of-CLEVR	SQOOP
question encoder	identity	LSTM
hidden dimension	128	128
layers	2	2
parameters	20k	118k

Table 6: Question-only model configuration for all experiments on Sort-of-CLEVR and SQOOP.

D TRAINING PROTOCOL

The optimisation settings are given in Table 13 and are held fixed across models and experimental conditions wherever possible. The only exceptions are baselines that failed to train under the shared protocol. In these cases, the optimisation settings were adjusted until training became reliable, with all deviations recorded in Table 14.

For each run, the checkpoint with the highest validation accuracy is selected and evaluated once on the test set. The validation set is always drawn from the same distribution as the corresponding training data. On Sort-of-CLEVR, the validation and test sets contain separate scenes sampled from the same distribution. On SQOOP, validation examples contain object pairings observed during training, while the test set contains held-out pairings that are not used for model selection.

Models are implemented in PyTorch and trained using bf16 mixed precision. Full dataset-generation settings are provided in Table 4, while the complete configuration of each model is given in Appendix C.

All runs used a single GPU, with an average training run taking roughly 45 minutes.

E ADDITIONAL RESULTS

E.1 SORT-OF-CLEVR BY SUBTYPE

The ternary gap on Sort-of-CLEVR is concentrated in the *Count obtuse* subtype, where SyncNet reaches 62.2% accuracy against 69.8%, 71.4% and 71.3% for the Transformer, Shared Workspace, and FiLM. The question asks how many third objects form an obtuse triangle with two queried objects, so information from several objects must be combined simultaneously. It is the hardest subtype for every model, so task difficulty contributes alongside the architectural restriction.

E.2 PHASE CAPACITY AND COMMUNICATION TIME

These sweeps vary one setting at a time around the main model. The architecture makes one quantitative prediction, and it concerns the phase dimension. At most d directions can be mutually orthogonal in d dimensions, which bounds how many senders a receiver can separate at once. Accuracy should therefore rise with d until there is enough capacity and flatten after that, and the gain should be largest where the most senders must be separated.

Setting	Sort-of-CLEVR (CNN+MLP)	SQOOP (Conv+LSTM)
encoder	shared, 128 channels	shared, 64 channels
question encoder	MLP	LSTM
classifier hidden dimension	256	256
parameters	6.11M	9.32M

Table 7: Convolutional baseline configuration for all experiments on Sort-of-CLEVR and SQOOP.

Setting	Sort-of-CLEVR	SQOOP
encoder	shared, 128 channels	shared, 64 channels
question encoder	identity	LSTM
g_θ hidden dimension	256	256
f_ϕ hidden dimension	256	256
pairing grid (max-pooled)	5×5	8×8
coordinate channels	yes	yes
parameters	0.44M	1.26M

Table 8: Relation Network configuration for all experiments on Sort-of-CLEVR and SQOOP.

Figure 4(a) shows this on Sort-of-CLEVR. Overall accuracy rises from 73.5% at $d = 2$ to 90.0% at $d = 3$, then sits between 91% and 92% through $d = 8$. The ternary family gains most, from 56.3% at $d = 2$ to 73.7% at $d = 3$ and a peak of 79.1% at $d = 4$, and seed variation is widest at $d = 2$. Two dimensions cannot hold the three orthogonal directions a three-way interaction needs, and the head must read from those three senders, giving four participants to separate. The peak at $d = 4$ matches that count, though the argument bounds the directions available rather than showing the model uses them.

SQOOP does not follow the pattern. Held-out accuracy is highest at $d = 2$ and then sits between 68.5% and 72.9%, with wide variation throughout. Its questions name two objects rather than three, so the demand on capacity is lower and the flat curve is what the argument predicts. Why $d = 2$ beats $d = 3$ it does not explain, since the head participates here, too. We report the peak without an account of it.

Internal steps behave the other way round. Little changes beyond $T = 2$ on Sort-of-CLEVR, while SQOOP rises from 57.8% at $T = 2$ to 73.0% at $T = 8$ before falling at $T = 16$, so SQOOP benefits from more time for communication to develop. The number of modules has little effect up to $M = 8$ on either task and degrades SQOOP thereafter, consistent with more senders sharing a phase space of fixed dimension.

F Ablation and override definitions

F.1 Ablations

Each variant changes exactly one component of the main model and is trained from scratch under the protocol of Appendix D. All other components are those of Section 3.

No spatial preference. The competition of Equation 7 assigns spatial cell p to module k by phase alignment with the module anchor, plus a learnt spatial preference $b_k(p)$ that encourages modules to claim distinct regions. The ablation removes it:

$$\text{no spatial preference:} \qquad b_k(p) = 0, \tag{21}$$

with $b_k(p)$ neither learnt nor used, leaving the assignment to emerge from phase alignment alone.

No structural modularity. In the main model each module has its own recurrent cell GRU_k in Equation 10. The ablation replaces these with a single cell shared across modules:

$$\text{no structural modularity:} \qquad h_k^{(t)} = \mathrm{GRU}\big([X_k; r_k^{(t)}], h_k^{(t-1)}\big) \quad \text{for all } k, \tag{22}$$

Setting	Sort-of-CLEVR	SQOOP
encoder	shared, 128 channels	shared, 64 channels
question encoder	identity	LSTM
residual blocks	4	4
block channels	128	128
classifier channels	512	512
classifier hidden dimension	1024	1024
coordinate channels	yes	yes
parameters	2.41M	3.26M

Table 9: FiLM configuration for all experiments on Sort-of-CLEVR and SQOOP.

Setting	Sort-of-CLEVR	SQOOP
encoder	patches (5×5)	patches (4×4)
question encoder	identity	token embedding, 32-d
hidden dimension	128	128
encoder layers	4	4
attention heads	4	4
feed-forward multiplier	2	4
dropout	0	0
positional encoding	learnt 2D	learnt 2D
question conditioning	token	broadcast-concatenate
readout	CLS	CLS
shared layer weights	no	no
parameters	0.63M	1.01M

Table 10: Transformer configuration for all experiments on Sort-of-CLEVR and SQOOP.

communication is unchanged. The shared cell is not widened to compensate, so this variant has fewer parameters than the main model.

Communication. The remaining ablations vary how communication is routed, combining an address source, phase or content, with a rule for combining messages, linear or softmax. The main model uses phase addresses with a linear read, Equation 12, so called because the messages are combined with unnormalised alignment weights rather than through a softmax. The three alternatives complete the combinations:

$$\text{phase + softmax:} \quad r_k^{(t)} = \sum_{j \neq k} \underset{j \neq k}{\mathrm{softmax}}\big(\tau \, \langle z_j^{(t-1)}, z_k^{(t-1)} \rangle\big) m_j^{(t)}, \tag{23}$$

$$\text{content + linear:} \quad z_k^{(t)} = \Pi(W_z h_k^{(t)}) \quad \text{at every step, replacing Equation 13,} \tag{24}$$

$$\text{content + softmax:} \quad r_k^{(t)} = \sum_{j \neq k} \omega_{kj} m_j^{(t)}, \qquad \omega_{kj} = \underset{j \neq k}{\mathrm{softmax}} \left(\tfrac{1}{\sqrt{d_h}} \langle W_Q h_k^{(t-1)}, W_K h_j^{(t-1)} \rangle \right), \tag{25}$$

where τ is a softmax temperature. The first two keep the shared channel and change only one choice each. The third also replaces the shared channel with direct attention between modules, so its addresses are asymmetric and it changes both the address source and the communication topology; *content + linear* is therefore the controlled comparison for the address source, as noted in Section 5.3.

F.2 Overrides

Overrides are applied to a trained model at test time. No parameter changes and no retraining take place, and each acts only on the module phases: content states $h_k^{(t)}$, module inputs X_k, the assignment weights $\alpha_{kp}^{(t)}$ and the anchors $\bar{a}_k$ are left as the trained model produces them. Perceptual modules are indexed $k = 1, \ldots, M$ and the head $k = 0$, as in Section 3. Each override is a parameter-free wrapper around the trained model, so the unmodified forward pass and the overridden one differ in exactly the lines below.

Setting	Sort-of-CLEVR	SQOOP
encoder	patches (5×5)	patches (4×4)
question encoder	identity	token embedding, 32-d
hidden dimension	128	128
layers (one shared layer, applied 4 times)	4	4
attention heads	4	4
feed-forward multiplier	2	2
question conditioning	token	token
readout	CLS	CLS
workspace slots	8	8
top-k fraction	0.2	0.2
key size	32	32
parameters	0.58M	0.71M

Table 11: Shared Workspace configuration for all experiments on Sort-of-CLEVR and SQOOP.

Setting	Sort-of-CLEVR	SQOOP
encoder	shared, 1×1 adapter to 64	shared, 64 channels
question encoder	identity	token embedding, 32-d
modules M	6	6
phase dimension d	6	6
internal steps T	8	8
step size γ	0.5	0.5
module state d_h	96	96
message width d_m	4	4
oscillators per cell	16, each 4-d (64 channels)	16, each 4-d (64 channels)
field steps T_x, step size γ_x	8, 1.0	8, 1.0
parameters	1.03M	0.83M

Table 12: SyncNet configuration for all experiments on Sort-of-CLEVR and SQOOP.

Freeze. After the binding stage, set

$$z_k^{(t)} = z_k^{(0)} \quad \text{for } k = 1, \ldots, M \text{ and } t = 1, \ldots, T, \tag{26}$$

discarding the output of Φ for those modules while retaining it for the head, whose phase evolves under Equation 13 as usual. The head is excluded deliberately: its initial phase is a random unit vector carrying no information about the scene, so a frozen head would read a fixed random projection of the channel at every step and the resulting collapse would measure a disabled readout rather than the perceptual modules' need for an evolving phase.

Zero. Set $z_k^{(t)} = e$ for every k including the head and every t, where e is a single fixed unit vector. Every inner product in Equation 12 is then 1 and each participant receives the unweighted average of the other messages. Addressing is removed; the messages are untouched.

Shuffle-phase. Draw a permutation π of $\{1, \ldots, M\}$ and, after the T_b binding iterations of Equation 7 and before the first internal step, replace

$$z_k^{(0)} \leftarrow z_{\pi(k)}^{(0)} \quad \text{for } k = 1, \ldots, M, \tag{27}$$

leaving X_k, $h_k^{(0)}$ and the head unchanged. Phases then evolve normally for all T steps. The set of phases and the angles between them are preserved exactly; what changes is which content state each phase is attached to.

Shuffle-binding. As above, but the permutation is applied to the bound pairs rather than the phases alone, so that X_k and $z_k^{(0)}$ are permuted together and module k receives module $\pi(k)$'s binding intact. The pairing of phase to content is preserved; what changes is which module holds which region.

Setting	Value
Optimiser	AdamW
Learning rate	3×10^{-4}
Weight decay	0.01
Schedule	2,000 warmup steps, then cosine to zero
Batch size	256
Gradient clipping	1.0
Optimisation steps	100,000

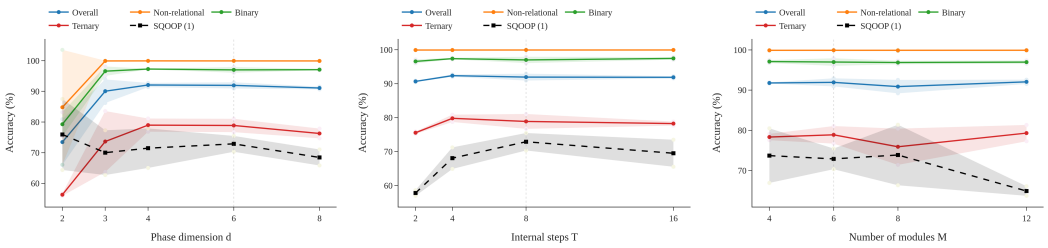
Table 13: The shared training configurations. Every reported result uses these settings except the rows listed in Table 14.

Model	Task	Effective batch	Learning rate	Weight decay	Steps
Transformer	SQOOP	—	10^{-4}	—	—
Workspace	SQOOP	—	10^{-4}	—	—
RelNet	SQOOP	2,048	—	—	20,000
Conv+LSTM	SQOOP	2,048	10^{-4}	0	40,000

Table 14: Deviations from the shared configuration, by model and task. The dash indicates shared value.

SYNCNET *Q-Only* CNN+MLP RELNET FILM TRANS. WORKS.

	SYNCNET	*Q-Only*	CNN+MLP	RELNET	FILM	TRANS.	WORKS.
Non-relational	99.9	50.7	71.7	99.9	100.0	100.0	100.0
Query shape	100.0	49.3	65.0	100.0	100.0	100.0	100.0
Left of centre	99.9	50.6	75.8	99.9	100.0	100.0	100.0
Top half	99.9	52.1	74.6	99.9	100.0	100.0	100.0
Binary	97.0	43.6	73.0	97.2	98.9	98.3	98.8
Closest shape	94.4	50.1	66.6	96.1	99.0	96.6	97.8
Furthest shape	96.6	50.4	74.0	95.5	97.6	98.5	98.6
Count same shape	99.9	29.9	78.7	100.0	99.9	100.0	100.0
Ternary	78.9	54.1	54.3	76.7	87.6	82.9	85.4
Count in box	88.9	69.6	69.0	87.4	97.3	91.1	94.0
On band	85.8	51.8	56.1	83.6	93.7	87.4	90.5
Count obtuse	62.2	40.5	37.3	58.7	71.3	69.8	71.4



(a) Phase dimension d. (b) Internal steps T. (c) Number of modules M.

Figure 4: Sweeps around the main model, varying one setting with all else fixed; the dotted line marks the value used in the main model. Sort-of-CLEVR accuracy is shown by question family alongside SQOOP held-out accuracy at pair variety 1.