Learning to Reason Iteratively and Parallelly for Complex Visual Reasoning Scenarios

Shantanu Jaiswal[1] **Debaditya Roy**[1] **Basura Fernando**[1,2] **Cheston Tan**
[1] Institute of High-Performance Computing,
Agency for Science, Technology and Research Singapore
[2] Centre for Frontier AI Research,
Agency for Science, Technology and Research Singapore

Abstract

Complex visual reasoning and question answering (VQA) is a challenging task that requires compositional multi-step processing and higher-level reasoning capabilities beyond the immediate recognition and localization of objects and events. Here, we introduce a fully neural *Iterative* and *Parallel Reasoning Mechanism* (IPRM) that combines two distinct forms of computation – iterative and parallel – to better address complex VQA scenarios. Specifically, IPRM's *"iterative"* computation facilitates compositional step-by-step reasoning for scenarios wherein individual operations need to be computed, stored, and recalled dynamically (e.g. when computing the query *"determine the color of pen to the left of the child in red t-shirt sitting at the white table"*). Meanwhile, its *"parallel"* computation allows for the simultaneous exploration of different reasoning paths and benefits more robust and efficient execution of operations that are mutually independent (e.g. when counting individual colors for the query: *"determine the maximum occurring color amongst all t-shirts"*). We design IPRM as a lightweight and fully-differentiable neural module that can be conveniently applied to both transformer and non-transformer vision-language backbones. It notably outperforms prior task-specific methods and transformer-based attention modules across various image and video VQA benchmarks testing distinct complex reasoning capabilities such as compositional spatiotemporal reasoning (AGQA), situational reasoning (STAR), multi-hop reasoning generalization (CLEVR-Humans) and causal event linking (CLEVRER-Humans). Further, IPRM's internal computations can be visualized across reasoning steps, aiding interpretability and diagnosis of its errors.

1 Introduction

Visual reasoning and question answering (VQA) at its core requires a model to identify relevant visual operations, execute those operations, and then combine their results to make an inference. Complex visual reasoning scenarios (depicted in fig. 1) are particularly challenging in this regard. They require models to reason compositionally over a large number of reasoning steps and to engage in a variety of higher-level reasoning operations such as causal linking, logical reasoning, and spatiotemporal processing that extend beyond core perception capabilities. In this context, two powerful computational priors exist – iterative and parallel. While each has its own limitations, when combined, they can complement each other and effectively address the challenges of complex VQA tasks. Specifically, **iterative computation**, wherein individual operations are identified and composed in a step-by-step manner, is a beneficial prior for multi-step reasoning scenarios explored by past VQA works [26, 23, 16]. However, pure iterative computation can exhibit limitations in scenarios wherein the entailed visual operations are independent of one-another, or where distinct stimuli need to be processed and tracked simultaneously.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Figure 1: Complex VQA scenarios (CLEVR-Humans [33], GQA [27], CLEVRER-Humans[49]), AGQA[18] and STAR[72]) wherein combination of iterative (step-by-step) computation (blue phrases) and parallel computation (orange phrases) can be beneficial for reasoning.

For example, consider the first scenario shown in fig. 1. When executing the language phrase *"maximum occurring shape"* (i.e. *"what shape appears the most"*), a purely iterative method would: (i) compute the count of each shape (each of which itself could take multiple iterations), (ii) then update and maintain the counts in memory (without forgetting count of all previous shapes), and (iii) finally, recall each shape's count to compute the *"maximum"* [1]. Besides taking more reasoning steps than required, such computation also increases the demand for information retention and recall in memory, which in this scenario could scale by the number of shapes to be counted. In complex video reasoning scenarios, a purely iterative method would similarly struggle in tracking and reasoning over multiple co-occurring events. In such scenarios, from both efficiency and efficacy perspectives, it is advantageous to process operations or stimuli in parallel, rather than solely iteratively.

More generally, **parallel computation** facilitates the simultaneous maintenance and exploration of distinct reasoning paths, and thereby enables reasoning to be more comprehensive, efficient and robust. For example, to compute *"maximum occuring shape"*, parallel compute can enable distinct shape queries to be simultaneously computed prior to computing the *"maximum"* operation. Similarly, it is effective for other scenarios illustrated in fig. 1 such as in processing independent logical clauses (*"{are X} and {Y made of plastic}"*) or when tracking and processing co-occurring events in videos.

Such computation can be implicitly realized in conventional transformer-based parallel attention mechanisms [64]. However, transformer-based attention does not explicitly incorporate iterative compositional computation [12, 39], which as described is beneficial for multi-step reasoning scenarios wherein operations need to be composed sequentially. Accordingly, while parallel computation may effectively compute the result of *"maximum occurring shape"* in fig. 1, it would potentially struggle to integrate the result with further operations such as *"green object with .."*, *"small object in front of green .."*, and *"color of .."* that need to be computed step-by-step to answer the question.

Based on the above insights, we design the **Iterative and Parallel Reasoning Mechanism (IPRM),** a novel neural reasoning architecture that combines step-by-step iterative computation with the ability to process multiple independent operations and stimuli simultaneously. Inspired by how humans utilize working memory [15, 7] to facilitate complex reasoning, IPRM internally maintains a latent memory of parallel "operation states", keyed to which are "results states". Given vision and language inputs, IPRM performs the following *iterative* computation. First, it forms a new set of *parallel* operations by retrieving relevant language information conditioned on its prior operation states. Then, it "executes" these operations *parallelly* by retrieving relevant visual information conditioned on its new operations as well as prior result states. Finally, it integrates its new operations (and their results) into memory by dynamically composing these operations with one-another as well as prior operation states, and subsequently, repeats the entire process in its next *iterative* step.

This strategy effectively enables us to take advantage of both parallel and iterative computations and notably helps improve state-of-arts across various complex image and video reasoning tasks using a single reasoning mechanism. Equally importantly, IPRM's internal computations can be visualized across reasoning steps, which helps better interpret what operations it was doing and accordingly where it was looking visually when processing a complex reasoning scenario.

[1] Assuming *"max"* is applied after counts of all shapes are computed.

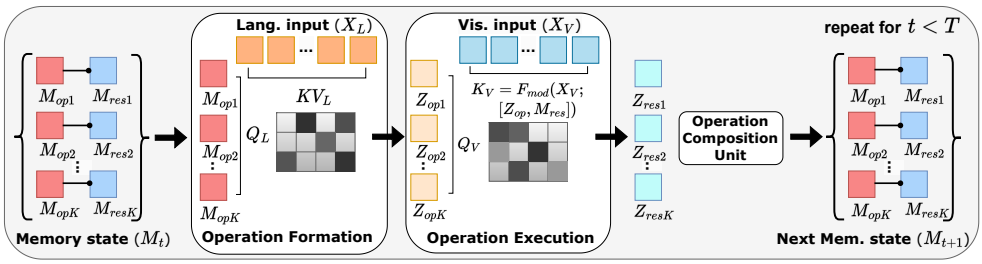

Figure 2: IPRM's computation flow diagram. First, a new set of K-parallel latent operations $\mathbf{Z_{op}}$ are retrieved from language features $\mathbf{X_L}$ conditioned on prior operation states $\mathbf{M_{op}}$. Then, visual features $\mathbf{X_V}$ are queried conditioned on both $\mathbf{Z_{op}}$ and prior result states results $\mathbf{M_{res}}$, to form the new results $\mathbf{Z_{res}}$. Finally, both $\mathbf{Z_{res}}$ and $\mathbf{Z_{op}}$ are passed to the Operation Composition Unit (see 2.3), the output of which becomes the new memory state $\mathbf{M}$.

2 Iterative and Parallel Reasoning Mechanism

Our proposed iterative-and parallel-reasoning mechanism (IPRM) is a fully-differentiable neural architecture. Given visual features $\mathbf{X_V} \in \mathbb{R}^{N_V \times D_V}$ and language or task-description features $\mathbf{X_L} \in \mathbb{R}^{N_L \times D_L}$, IPRM outputs a *"reasoning result"* $\mathbf{y_s} \in \mathbb{R}^{D_m}$ and, optionally, a set of *"reasoning result tokens"* $\mathbf{Y_R} \in \mathbb{R}^{N_m \times D_m}$. As previously mentioned, IPRM operates iteratively for T reasoning steps and internally, maintains an explicit memory $\mathbf{M} : \{\mathbf{M_{op}}, \mathbf{M_{res}}\}$. The memory is modelled as a set of *"operation states"* $\mathbf{M_{op}} \in \mathbb{R}^{N_{op} \times D_m}$, keyed to which are *"result states"* $\mathbf{M_{res}} \in \mathbb{R}^{N_{op} \times D_m}$ as shown in fig. 2. Here, N_{op} denotes the number of parallel operations to be computed while D_m denotes the mechanism's internal feature dimension. On a high level, at each reasoning step (denoted by $t \in \{1, \cdots, T\}$), IPRM performs the following computations:

1. First, conditioned on the existing operation states $\mathbf{M_{op,t}}$, we retrieve relevant information from language or task-description features $\mathbf{X_L}$ to form a new set of latent operations $\mathbf{Z_{op,t}} \in \mathbb{R}^{N_{op} \times D_m}$. We term this computation as *"Operation Formation"*.

$$\mathbf{Z_{op,t}} = \mathbf{Op\_Form}(\mathbf{X_L}; \mathbf{M_{op,t}}) \tag{1}$$

2. Then, conditioned on the latent operations $\mathbf{Z_{op,t}}$ and the existing result state $\mathbf{M_{res,t}}$, we attend and retrieve relevant information from visual features $\mathbf{X_V}$ which represents a new set of latent results $\mathbf{Z_{res,t}} \in \mathbb{R}^{N_{op} \times D_m}$ corresponding to $\mathbf{Z_{op,t}}$. We term this computation as *"Operation Execution"*.

$$\mathbf{Z_{res,t}} = \mathbf{Op\_Exec}(\mathbf{X_V}; [\mathbf{Z_{op,t}}, \mathbf{M_{res,t}}]) \tag{2}$$

3. Finally, to facilitate interaction amongst parallel operations, we perform inter-operation attention. Here, each operation $\mathbf{Z_{op\_k,t}}; \mathbf{k} \in \{1, \cdots, N_{op}\}$, is composed with other operations in $\mathbf{Z_{op,t}}$ as well as prior operation states $\mathbf{M_{op[t-W:t]}}$ within a lookback-window W. The corresponding result $\mathbf{Z_{res\_k,t}}$ is similarly composed with other results $\mathbf{Z_{res,t}}$ and prior result states denoted as $\mathbf{M_{res[(t-W):t]}}$. We term this computation as *"Operation Composition"*

$$\mathbf{M_{t+1}} = \mathbf{Op\_Comp}(\{\mathbf{Z_{op,t}}, \mathbf{Z_{res,t}}\}, \mathbf{M_{[(t-W):t]}}) \tag{3}$$

   As shown in eq. (3), this output is the new memory state $\mathbf{M_{t+1}} : \{\mathbf{M_{op}}, \mathbf{M_{res}}\}$.

The overall computation flow is illustrated in fig. 2, and we provide specific details and intuitions behind these computations in the following sub-sections.

2.1 Operation Formation

The *"operation formation"* stage conceptually models a reasoner that based on its prior set of operations, decides what language features to retrieve in order to form the next set of relevant operations. This can be effectively implemented through conventional attention mechanisms. Specifically,

the cumulative set of prior operations (maintained in $\mathbf{M_{op,t}}$) can be projected to form the 'query' $\mathbf{Q_{L,t}} \in \mathbb{R}^{N_{op} \times D_m}$ representing "what features to look for". The language features $\mathbf{X_L}$ can be projected to form the 'key' $\mathbf{K_L} \in \mathbb{R}^{N_L \times D_m}$ and 'value' $\mathbf{V_L} \in \mathbb{R}^{N_L \times D_m}$. Finally, the new set of latent operations $\mathbf{Z_{op,t}}$ can be retrieved by computing $\texttt{attn}(\mathbf{Q_L}, \mathbf{K_L}, \mathbf{V_L})$. These steps are formally represented below:

$$\mathbf{Q_{L,t}} = \mathbf{W_{L,q2}}(\texttt{Tanh}(\mathbf{W_{L,q1}}(\mathbf{M_{op,t}}))), \mathbf{K_L} = \mathbf{W_{L,k}}(\mathbf{X_L}), \mathbf{V_L} = \mathbf{W_{L,v}}(\mathbf{X_L}) \tag{4}$$

$$\mathbf{Z_{op,t}} = \texttt{attn}(\mathbf{Q_{L,t}}, \mathbf{K_L}, \mathbf{V_L}) \tag{5}$$

Here, $\mathbf{W_{L,q2}} \in \mathbb{R}^{D_m \times D_m}$, $\mathbf{W_{L,q1}} \in \mathbb{R}^{D_m \times D_m}$, $\mathbf{W_{L,k}} \in \mathbb{R}^{D_m \times D_l}$ and $\mathbf{W_{L,v}} \in \mathbb{R}^{D_m \times D_l}$. Note $\mathbf{K_L}$ and $\mathbf{V_L}$ are not computation-step dependent and only computed once. We use a simple linear-modulated formulation (with appropriate broadcasting and projection weight $\mathbf{W_a} \in \mathbb{R}^{D_k \times 1}$) to implement `attn(.)` (further details in appendix sec. 13).

2.2 Operation Execution

In the *"operation execution"* stage, the reasoner determines what visual features need to be retrieved depending on both the newly formed operations and existing result states. To model the constituent visual attention mechanism, we draw insights from existing recurrent visual reasoning methods [26, 63] that incorporate feature modulation for memory-guided attention. Specifically, we retrieve a set of feature modulation weights $\mathbf{S_{V,t}} \in \mathbb{R}^{N_{op} \times D_m / r}$ through a joint projection of the new operations $\mathbf{Z_{op,t}}$ and prior results $\mathbf{M_{res,t}}$ as shown in eq. (6).

$$\mathbf{S_{V,t}} = \mathbf{W_{V,s}}([\mathbf{W_{V,op}}(\mathbf{Z_{op,t}}), \mathbf{W_{V,res}}(\mathbf{M_{res,t}})]) \tag{6}$$

Here, r is a feature reduction ratio [21, 29]. $\mathbf{S_{V,t}}$ is then applied dimension wise to a projection of $\mathbf{X_V}$ to retrieve an intermediate attention key $\mathbf{K'_{V,t}} \in \mathbb{R}^{N_{op} \times N_k \times D_m / r}$. The final attention key $\mathbf{K_{V,t}}$ is then obtained through a joint multi-layer-projection of $\mathbf{K'_{V,t}}$ and the previously projected representation of $\mathbf{X_V}$ as shown in eq. (7).

$$\mathbf{K'_{V,t}} = \mathbf{S_{V,t}} \odot \mathbf{W_{V,k1}}(\mathbf{X_V}),\ \mathbf{K_{V,t}} = \mathbf{W_{V,k3}}(\phi(\mathbf{W_{V,k2}}([\mathbf{W_{V,k1}}(\mathbf{X_V}), \mathbf{K'_{V,t}}]))) \tag{7}$$

Finally, the attention query and value are formed through separate projections of $\mathbf{Z_{op,t}}$ and $\mathbf{X_V}$ respectively. These are then fed together with $\mathbf{K_{V,t}}$ to the attention function to retrieve the new operation results $\mathbf{Z_{res,t}}$ as shown in eq. (8). Intuitively, the overall process allows for both prior results and the new set of operations to jointly guide visual attention.

$$\mathbf{Q_{V,t}}, \mathbf{V_{V,t}} = \mathbf{W_{V,q}}(\mathbf{Z_{op,t}}), \mathbf{W_{V,v}}(\mathbf{X_V}), \mathbf{Z_{res,t}} = \texttt{attn}(\mathbf{Q_{V,t}}, \mathbf{K_{V,t}}, \mathbf{V_{V,t}}) \tag{8}$$

Here, $\mathbf{W_{V,op}} \in \mathbb{R}^{D_m/r \times D_m}$, $\mathbf{W_{V,res}} \in \mathbb{R}^{D_m/r \times D_m}$, $\mathbf{W_{V,s}} \in \mathbb{R}^{D_m/r \times 2D_m/r}$, $\mathbf{W_{V,k1}} \in \mathbb{R}^{D_m/r \times D_v}$, $\mathbf{W_{V,k2}} \in \mathbb{R}^{D_m/r \times 2D_m/r}$, $\mathbf{W_{V,k3}} \in \mathbb{R}^{D_m/r \times D_m/r}$, $\mathbf{W_{V,q}} \in \mathbb{R}^{D_m/r \times D_m}$ and $\mathbf{W_{V,v}} \in \mathbb{R}^{D_m \times D_m}$. We set ϕ to ELU for vision backbones.

2.3 Operation Composition

Finally, in the *"operation composition"* stage, the reasoner first integrates the executed operations $\mathbf{Z_{op,t}}$ and their results $\mathbf{Z_{res,t}}$ into the existing memory state $\mathbf{M_t}$ through a simple recurrent update as shown in eqs. (9) and (10). Then, to mitigate redundancy amongst parallel operations and to retrieve relevant knowledge from prior-step operations, it dynamically composes individual operation states $\mathbf{M'_{op,t+1}}$ with other operation states in $\mathbf{M'_{op,t+1}}$ and also prior operation states in $\mathbf{M_{op,t-W:t}}$. Here, W is an attention look-back window.

This composition is achieved through computing inter-operation attention as illustrated in fig. 3. Specifically, $\mathbf{M'_{op,t+1}}$ is projected to obtain a set of queries $\mathbf{Q_{op,t}}$, while the token-wise concatenation of $\mathbf{M'_{op,t+1}}$ and $\mathbf{M_{op,t-W:t}}$ are projected to obtain the operation attention keys $\mathbf{K_{op,t}}$ and values $\mathbf{V_{op,t}}$. A second set of values $\mathbf{V_{res,t}}$ are also formed through projection of respective result states as shown in eq. (13). Further, an identity attention mask $\mathbf{I_{N_{op}}}$ is used to ensure that operations in $\mathbf{Q_{op,t}}$,

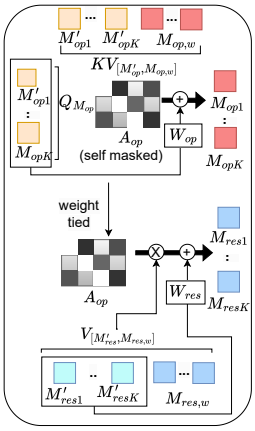

Figure 3: Operation Composition Unit

can only attend to other operations and not themselves. This is done to enable a higher degree of operation composition. As shown in eq. (14), $\mathbf{Q_{op,t}}$, $\mathbf{K_{op,t}}$, $\mathbf{V_{op,t}}$ and $\mathbf{I_{N_{op}}}$ are passed to the attention operation, which outputs an intermediate representation $\mathbf{M''_{op,t+1}}$ and the softmaxed-attention weights $\mathbf{A_{op,t}}$. $\mathbf{M''_{op,t+1}}$ is then added to a projection of $\mathbf{M'_{op,t+1}}$ to effectively combine attended operation states with the original operation states, and thereby form the next mem. operation state $\mathbf{M_{op,t+1}}$.

Finally, the next result states are obtained by applying $\mathbf{A_{op,t}}$ on $\mathbf{V_{res,t}}$ and then adding a projection of $\mathbf{M'_{res,t+1}}$ as shown in eq. (16). Note $\mathbf{A_{op,t}}$ is specifically utilized to ensure that results are composed based on attentions between operation states. Here, all the mentioned weights $\mathbf{W_{..}} \in \mathbb{R}^{D_m \times D_m}$ and $[\cdot;\cdot]$ represents token-wise concatenation. $\mathbf{I_{N_{op}}}$ in eq. (14) is identity matrix which is concatenated with zeros (i.e. unmasked) for window tokens if window len. $w > 0$.

$$\mathbf{M'_{op,t+1}} = \mathbf{W_{op,U}}(\mathbf{Z_{op,t}}) + \mathbf{W_{op,H}}(\mathbf{M_{op,t}}) \tag{9}$$

$$\mathbf{M'_{res,t+1}} = \mathbf{W_{res,U}}(\mathbf{Z_{res,t}}) + \mathbf{W_{res,H}}(\mathbf{M_{res,t}}) \tag{10}$$

$$\mathbf{Q_{op,t}}; \mathbf{K_{op,t}} = \mathbf{W_{op,q}}(\mathbf{M'_{op,t+1}}); \mathbf{W_{op,k}}([\mathbf{M'_{op,t+1}}; \mathbf{M_{op,t-W:t}}]) \tag{11}$$

$$\mathbf{V_{op,t}} = \mathbf{W_{op,v}}([\mathbf{M'_{op,t+1}}; \mathbf{M_{op,t-W:t}}]) \tag{12}$$

$$\mathbf{V_{res,t}} = \mathbf{W_{res,v}}([\mathbf{M'_{res,t+1}}; \mathbf{M_{res,t-W:t}}]) \tag{13}$$

$$\mathbf{M''_{op,t+1}}, \mathbf{A_{op,t}} = \texttt{attn}(\mathbf{Q_{op,t}}, \mathbf{K_{op,t}}, \mathbf{V_{op,t}}, \text{mask=}\mathbf{I_{N_{op}}}) \tag{14}$$

$$\mathbf{M_{op,t+1}} = \mathbf{M''_{op,t+1}} + \mathbf{W_{op,u2}}(\mathbf{M'_{op,t+1}}) \tag{15}$$

$$\mathbf{M_{res,t+1}} = \mathbf{A_{op,t}}(\mathbf{V_{res,t}}) + \mathbf{W_{res,v2}}(\mathbf{M'_{res,t+1}}) \tag{16}$$

Obtaining Reasoning Summary As mentioned before, our proposed mechanism outputs a set of *"reasoning result tokens"* $\mathbf{Y_R}$ and a *"reasoning result"* $\mathbf{y_s}$. $\mathbf{Y_R}$ is simply equivalent to the last memory result states $\mathbf{M_{res,T+1}}$. To obtain $\mathbf{y_s}$, we perform attention on the last operation states $\mathbf{M_{op,T+1}}$ by utilizing a summary representation $\mathbf{l_s} \in \mathbb{R}^{D_l}$ of $\mathbf{X_L}$ as the attention-query.

We set $\mathbf{l_s}$ to be the first token in case of transformer-based language backbones and as last hidden state in case of LSTM-based language backbones. As shown in eq. (17), $\mathbf{l_s}$ is projected to obtain a single-token attention query $\mathbf{p_q}$ while $\mathbf{M_{op,T+1}}$ is projected to obtain the attention keys $\mathbf{P_k}$. The attention value is simply the result states $\mathbf{M_{res,T+1}}$, and the output of the attention function is the *"reasoning result"*. Intuitively, this computation corresponds to the reasoner deciding which final operation states in $\mathbf{M_{op,T+1}}$ are most relevant to the summary of the input language or task-description $\mathbf{X_L}$, based on which corresponding result states $\mathbf{M_{res,T+1}}$ are weighted and retrieved.

$$\mathbf{p_q}, \mathbf{P_k} = \mathbf{W_{pq,q}}(\mathbf{l_s}), \mathbf{W_{pk,k}}(\mathbf{M_{op,T+1}}) \tag{17}$$

$$\mathbf{y_s} = \texttt{attn}(\mathbf{p_q}, \mathbf{P_k}, \mathbf{M_{res,T+1}}) \tag{18}$$

Here, $\mathbf{W_{pq,q}} \in \mathbb{R}^{D_m \times D_l}$ and $\mathbf{W_{pk,k}} \in \mathbb{R}^{D_m \times D_m}$.

Reasoning mechanism general applicability. Our proposed iterative and parallel reasoning mechanism is an end-to-end trainable neural module. It can be conveniently applied on top of different vision and language backbones, and be trained directly as a new computational block with no specific adjustments. Further, IPRM is weight-tied which means that its number of parameters is constant regardless of number of computation steps and parallel operations. We provide parameter and computational details along with module implementation in appn. C.

Finally, while we propose and explore IPRM in context of complex VQA, we note that it can be interpreted as a general reasoning mechanism applicable for reasoning tasks beyond visual reasoning and question answering. Simply, its inputs $\mathbf{X_L}$ can be interpreted as reasoning task specification (e.g. question, task details, entailment statement, etc.) while $\mathbf{X_V}$ can be interpreted as reasoning stimuli (e.g. images, embodied scenes, language passages, etc). The reasoning process can then implicitly operate iteratively and parallelly as described to obtain reasoning outputs $\mathbf{y_s}$ and $\mathbf{Y_R}$.

Table 1: Comparison of IPRM with videoQA methods on STAR (left) and AGQAv2 (right). All methods operate on 32 frames unless otherwise mentioned in ().

Model	Int.	Seq.	Pred.	Feas.	Avg.
All-in-One [66]	47.5	50.8	47.7	44.0	47.5
Temp[ATP][5]	50.6	52.8	49.3	40.6	48.3
MIST [16]	55.5	54.2	54.2	44.4	51.1
InternVideo(8) [69]	62.7	65.6	54.9	51.9	58.7
SeViLA-BLIP2 [79]	63.7	70.4	63.1	62.4	64.9
Concat-Att-4L	60.2	66.9	70.8	64.7	64.9
Cross-Att-4L	60.0	67.2	68.9	68.4	65.0
IPRM(16)	62.9	70.0	**76.9**	67.1	68.1
IPRM	**64.2**	**72.9**	75.3	**69.1**	**69.9**

Metric	HCRN[37]	AIO[66]	Temp[5]	MIST[16]	GF [3]	IPRM
obj-rel	40.3	48.3	50.2	51.7	55.0	**57.0**
superlative	33.6	37.5	39.8	42.1	44.6	**47.4**
sequencing	49.7	49.6	48.3	67.2	53.2	**74.3**
exist	50.0	50.8	51.8	60.3	59.1	**61.9**
duration	43.8	45.4	49.6	**54.6**	52.8	51.3
act. recog.	5.5	19.0	19.0	19.7	14.2	**23.1**
open	36.3	-	-	50.6	56.1	**57.9**
binary	48.0	-	-	58.3	54.2	**61.6**
all	42.1	48.6	49.8	54.4	55.1	**59.9**

3 Experiments

We evaluate IPRM on STAR[72], AGQAv2[18] and CLEVRER-Humans[49] for video reasoning tasks and CLEVR-Humans[33], GQA [27] and CLEVR-CoGenT [32] for image reasoning tasks. For all tasks, we set IPRM's parallel operations (N_{op}) to 6, reasoning steps (T) to 9, reduction ratio (r) to 2 and window length (W) to 2 (informed by ablative analysis detailed in section 3.3). We follow task-specific practices (detailed in app.C.1) for respective vision and language backbones in our primary experiments, besides also demonstrating integration of IPRM with large-scale VL backbones such as CLIP. Further, besides task-specific methods, we also consider two prominent transformer-based VL modules as baselines – concat-att (where lang. and vis. tokens are concatenated as in [34, 35]) and cross-att (where lang. tokens are "query" to vis. tokens as "key" and "value"; as in [41, 1]). Further implementation and training details are provided in app.C.[2]

3.1 Video Reasoning and Question Answering (STAR, AGQAv2 and CLEVRER-Humans)

We first evaluate IPRM on recent video reasoning benchmarks. STAR [72] and AGQAv2 comprise real-world videos and test multiple reasoning skills in context of situational reasoning and compositional spatiotemporal reasoning respectively. STAR contains 60K questions testing four broad types of video reasoning abilities: *feasibility*, *interaction*, *prediction* and *sequence*. Meanwhile, AGQAv2 contains 2.27M balanced questions distributed across 16 different question types. As shown in table 1, IPRM obtains 69.9% average acc. on STAR and 59.9% overall acc. on AGQAv2, outperforming prior videoQA-specific methods by 5% on both benchmarks.

Interestingly, on STAR IPRM obtains a 12.2% and 6.7% improvement over SeViLA-BLIP2 on the predictive and feasability scenarios. This is possibly due to IPRM's capability to reason over multiple events simultaneously across frames which may help it better retrieve and cumulatively reason on relevant information to predict future events and determine feasible actions. Similarly, on AGQAv2, IPRM improves performances across various question types, notably achieving a 7% improvement in questions that require determining sequence of events. Further, IPRM also outperforms both 4-layer concat- and cross-attention modules on STAR (scaling further attention-layers was not found to benefit performance as detailed in appendix table 6).

Next, we evaluate IPRM on the CLEVRER-Humans benchmark [49], which comprises synthetic videos of simultaneous object motions and multiple collisions, and tests a model's ability to determine causal links between events. We perform zero-shot, finetuned and from-scratch evaluation. As shown in table 2, IPRM outpeforms task-specific neurosymbolic models NS-DR and VR-DP as well as state-of-the-art ALOE across the three settings. Specifically, IPRM improves zero-shot per-question acc. by 7%, finetuned per-question acc. by 18.8% and scratch per-question acc. by 6.2%. These results further suggest that IPRM can better track and process co-occuring events, and in this case, more accurately determine causal links.

[2]Experiments source code and checkpoints will be released publicly.

Table 2: Comparison of methods for CLEVRER-Humans [49] (Opt. is per option acc. and Qs. is per question acc.). IPRM achieves state-of-art across settings.

Model	Zero-shot		Finetune		Scratch	
	Opt.	Qs.	Opt.	Qs.	Opt.	Qs.
NS-DR[77]	51.0	32.0	-	-	-	-
VRDP[11]	50.9	31.6	-	-	-	-
CNNLSTM[49]	50.3	30.0	51.7	34.2	51.5	30.8
CNNBERT[49]	52.9	32.0	52.0	30.2	50.1	30.4
ALOE[10]	54.0	26.9	51.8	31.7	52.7	32.1
IPRM	**61.7**	**38.9**	**74.1**	**53.0**	**62.0**	**38.3**

Table 3: Comparison of methods on CLEVR-Humans (zero-shot and finetuned setting) and CLEVR-CoGenT (ValA is in-domain; ValB is out-of-domain setting). IPRM achieves state-of-art and does not require any extra supervision.

Model	Extra supv.	CLV-Hum		CLV-CoGen	
		ZS	FT	ValA	ValB
PG+EE [33]	Programs	54.0	66.6	96.6	73.7
NS-VQA [78]	Programs	-	67.8	**99.8**	63.9
FiLM [55]	None	56.6	75.9	98.3	78.8
MAC [26]	None	57.4	81.5	99.0	78.3
MDETR [34]	Bound. Box	59.9	81.7	**99.8**	76.7
IPRM	None	**63.8**	**85.5**	99.1	**80.3**

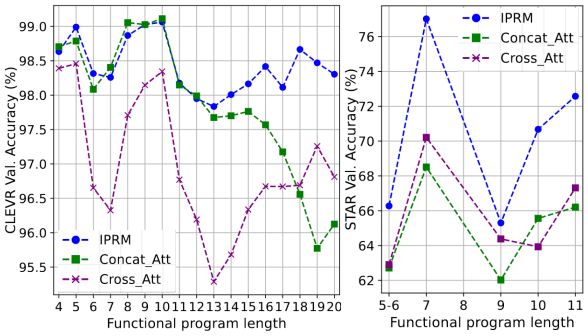

Figure 4: Acc. of IPRM (blue) across program lengths for CLEVR (left) and STAR (right). IPRM has significantly higher accs. at longer program lengths.

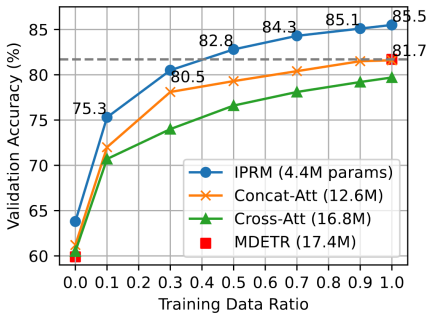

Figure 5: IPRM performance on CLEVR-Humans at different training data ratios of Cross- and Concat-Att.

3.2 Compositional Image Reasoning (CLEVR-Humans, CLEVR-CoGenT and GQA)

Here, we evaluate IPRM on challenging compositional image reasoning benchmarks. CLEVR-Humans tests generalization of multi-hop reasoning to free-form human crowdsourced questions which entail reasoning skills/scenarios beyond a model's training on the original CLEVR dataset and provides a limited finetuning set (2.5% of CLEVR). Similarly, CLEVR-CoGenT tests generalization on novel attribute compositions not observed in training (e.g."*gray cubes*" and "*red cylinders*" are in training but eval. is on "*gray cylinders*" and "*red cubes*"; see suppl. for exact specification).

As shown in table 3, IPRM achieves 3.9% and 3.8% improvements in zero-shot and fully-finetuned performance over prior state-of-art vision-language model MDETR. Further, IPRM neither requires bounding-box pre-training supervision (as done in MDETR) nor functional programs, and can be trained directly with only vision-language inputs and task supervision. fig. 5 illustrates IPRM's performance across different training ratios compared to MDETR and cross- and concat-att transformer modules. Notably, IPRM exceeds MDETR's fully-finetuned performance by 1.1% with only half of training data. Further, IPRM exhibits these improvements while being relatively lightweight (4.4M params) in comparison to MDETR's transformer blocks (17.4M; $\sim$4x more parameters) and cross- and concat-VL attention methods (16.8M and 12.6M respectively). These results suggest that IPRM exhibits strong generalization capabilities and more sample-efficient learning of novel reasoning skills and scenarios in context of multi-step imageQA.

For CLEVR-CoGenT, IPRM achieves state-of-art results in out-of-domain generalization on novel attribute compositions and outperforms MDETR (having parallel transformer compute) by 3.6% and MAC (iterative method) by 2%. This suggests the combination of iterative and parallel computation as done in IPRM can implicitly enable more disentangled feature processing and thereby improve compositional learning of primitive attributes in context of multi-step imageQA.

We also evaluate IPRM on the GQA benchmark which tests compositional VQA on real-world images. We perform comparisons with prior VQA methods as well as large-scale VL models such as LXMERT and 12-in-1 that do not utilize the ground truth scene graphs in GQA. As shown in table 4, IPRM achieves 60.3% , outperforming prior reasoning methods such as MCAN and LCGN as well as large-scale VL models such as LXMERT and 12-in-1. However, IPRM's performance is behind the

Table 4: Performances on GQA of imageQA methods that do not utilize ground-truth scene graphs. * indicates large-scale pretrained VL model

	MAC[26]	LCGN [23]	MCAN[80]	LXMERT*[61]	12-in-1*[47]	OSCAR*[44]	IPRM
GQA	53.9	55.8	57.4	60.0	60.0	**61.6**	60.3

larger VL model OSCAR which performs pretraining on 4.3M data samples collated from COCO, VG, SBU and Flickr. Note, IPRM is a standalone reasoning module only trained on GQA balanced with no pre-training. Finally, in appendix B.3, we also show how IPRM can more effectively improve performances of VL backbones such as CLIP on complex reasoning benchmarks in comparison to traditional cross- and concat-attention modules.

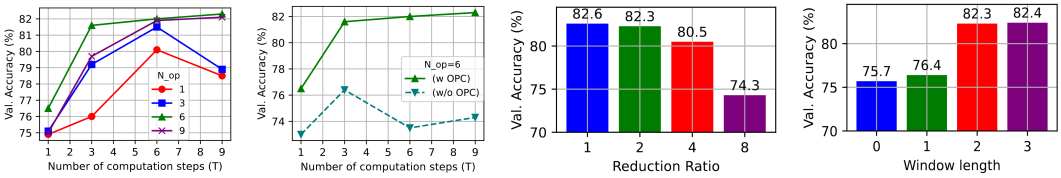

Figure 6: IPRM Model ablations in order: **(i)** Impact of number of parallel operations (N_{op}) vs computation steps (T). **(ii)** Impact of Operation Composition Block (OPC). **(iii)**: Impact of reduction ratio (r) and **(iv)** memory window length (W).

3.3 Model ablations and reasoning visualization

We perform ablations to study the contributions of IPRM's salient components. First, we analyze the impact of varying number of parallel operations (N_{op}) against number of iterative computation steps (T). We compare models with $N_{op} \in \{1, 3, 6, 9\}$ and $T \in \{1, 3, 6, 9\}$, resulting in 16 different models. Given the large amount of ablative models, we perform analysis on a reduced-resolution setting of CLEVR-Humans with pretraining on CLEVR for 15 epochs. As shown in fig. 6 (plot (i)), we find that T and N_{op} appear to be co-dependent and that neither by itself can lead to high performance. E.g. setting T = 1 generally results in peformances around 75% regardless of N_{op}, while setting N_{op}= 1 or 3 results in a sharp performance drop of 3% when changing T to 9 from 6. In contrast, for $N_{op} > 3$, we observe that performance increases steadily with higher T, suggesting that a higher number of parallel operations may prevent overfitting in a model with high computation steps. Overall, we find that ($N_{op} = 6, T = 9$) and ($N_{op} = 9, T = 9$) are the two best performing models achieving above 82% accuracy (with the former preferred as $N_{op} = 6$ performs better for diff. T compared to $N_{op} = 9$).

Next, we study the impact of the operation composition block (OPC) by evaluating $N_{op} = 6$ (and diff. computation steps T) with and without OPC. As shown in fig. 6 (plot (ii)), while $T = 1$ has a relatively low drop of $\sim$2%, the performance drops are more significant for higher T. The ($N_{op} = 6$, $T = 9$) model which reached $\sim$82% acc. with OPC, drops to $\sim$74% acc. without OPC.

Finally, we study the impacts of the dimension reduction ratio (r) and memory-lookback window length (W). As shown in fig. 6 (plot (iii)), $r = 2$ leads to negligible drop (0.3%) in performance compared to no dimension reduction (i.e. $r = 1$). However, setting r to 8 significantly deteriorates performance. Similarly, as shown in fig. 6 (plot (iv)), the optimal setting of window-length W is 2, and decreasing it below that significantly impacts performance.

In fig. 4, we assess the performance of IPRM across different functional program lengths (a proxy for reasoning steps) on the CLEVR and STAR benchmarks. As shown, IPRM maintains a high performance across both short and longer reasoning steps and for both image and video VQA scenarios. We also provide a condensed illustration of IPRM's reasoning visualization in fig. 7. Here, for brevity, we only show the model's visual attention for a subset of parallel operations and computation steps (see suppl. for full visualization). The left example ("what shape .. max occuring color") illustrates the model's usage of both iterative and parallel computation. At the first reasoning step, the model appears to be doing parallel operations to both identify "large cylinder" and compute "max occuring color". In the next step, it appears to have found "max occuring color" (cyan) and seems to check which of the two large cylinders match the color. Next, it highlights the correct

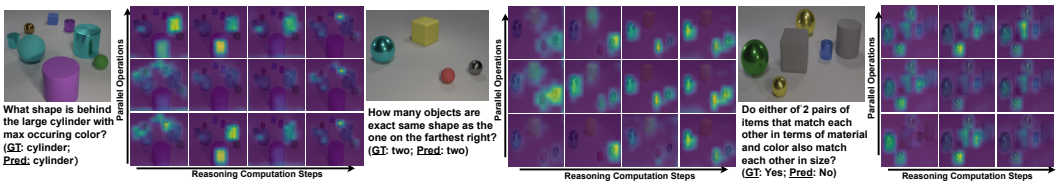

Figure 7: **Condensed reasoning visualization of IPRM**. In the first two examples, IPRM correctly utilizes both parallel and iterative computation to arrive at the correct answer. In the right example, IPRM seems to misunderstand question and outputs wrong ans. with less relevant attentions. See appendix for complete reasoning visualization and supplemental for CLIP integrated visualization on GQA.

cylinder and in the final step, it locates and predicts the correct object behind. The middle example in fig. 7 similarly shows computation for a question involving spatial ("farthest right"), similarity ("same shape") and counting ("how many") operations. Finally, the right example is an error-case where the model seems to incorrectly understand or execute requisite operations, thereby not producing expected attentions for "items with matching material and color" throughout steps.

4 Related Work

Visual reasoning methods and vision-language models. Multiple prior works have introduced effective visual reasoning methods in context of image and video question answering [33, 50, 2, 78, 13, 48, 63, 51, 30]. Prominent works include NMN[2], FILM [55], NSM [25], MAC [26], MCAN [80], NS-VQA [78], ALOE [10], VR-DP [11], SHG-VQA [62] and MIST [16]. In contrast to these works that show applicability of methods for particular VQA benchmarks/tasks, our work explores a more general direction of integrating parallel and iterative computation in a reasoning framework that we show to be effective for multiple complex VQA benchmarks and reasoning scenarios. More recently, vision-language models [42, 67, 35, 61, 43, 44, 68] and multimodal large-language-models [52, 45] with transformer-based mechanisms have shown impressive reasoning capabilities at scale. Notable examples include CLIP [56], MDETR [34], LXMERT [61], VinVL [81], BLIP [42, 41], Flamingo [1], LlavA [45], BEiT [68] and All-in-One [66]. We believe our work is complimentary to these developments, as it contributes an alternative and possibly more effective reasoning mechanism that can be integrated with such models in future to enhance complex VQA capabilities.

Memory and recurrence-augmented transformers. Multiple works have identified limitations of purely feedforward computation as realized in transformers and worked on encoding recurrence [28, 24, 9, 65] and memory-augmented computation [73, 6]. Notably, Recurrent Memory Transformer [6] and MemFormer[74] introduce recurrent and dynamic memory to improve language modelling capabilities. More recently, EMAT [75] introduces efficient memory to augment knowledge retrieval, MemViT [73] introduces a cache memory to retain prior context for long-video tasks, and [65] introduces memory for more effective action anticipation. While these methods study recurrent and memory-augmented computation on specific natural language processing and computer vision tasks, our work focuses on the integration of iterative-parallel computation and working memory in a single neural reasoning mechanism beneficial for complex VQA scenarios.

5 Conclusion

We introduced a novel fully-differentiable and end-to-end trainable iterative and parallel reasoning mechanism (IPRM) to address complex VQA scenarios. We comprehensively evaluated IPRM on various complex image and video VQA benchmarks testing distinct reasoning capabilities, and found it improves state-of-arts on multiple such benchmarks. We also performed quantiative ablations to study individual impacts of parallel and iterative computation besides qualitative analysis of IPRM's reasoning computation visualization.

Acknowledgment This research/project is supported by the National Research Foundation, Singapore, under its NRF Fellowship (Award# NRF-NRFF14-2022-0001). This research is also supported by funding allocation to B.F. by the Agency for Science, Technology and Research (A*STAR) under its SERC Central Research Fund (CRF), as well as its Centre for Frontier AI Research (CFAR).

References

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[2] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*, 2016.

[3] Z. Bai, R. Wang, and X. Chen. Glance and focus: Memory prompting for multi-event video question answering. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] B. Bogin, S. Subramanian, M. Gardner, and J. Berant. Latent compositional representations improve systematic generalization in grounded question answering. *Transactions of the Association for Computational Linguistics*, 9:195–210, 2021.

[5] S. Buch, C. Eyzaguirre, A. Gaidon, J. Wu, L. Fei-Fei, and J. C. Niebles. Revisiting the" video" in video-language understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2917–2927, 2022.

[6] A. Bulatov, Y. Kuratov, and M. Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

[7] A. Capon, S. Handley, and I. Dennis. Working memory and reasoning: An individual differences perspective. *Thinking & Reasoning*, 9(3):203–244, 2003.

[8] Y. Cong, W. Liao, H. Ackermann, B. Rosenhahn, and M. Y. Yang. Spatial-temporal transformer for dynamic scene graph generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16372–16382, 2021.

[9] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[10] D. Ding, F. Hill, A. Santoro, M. Reynolds, and M. Botvinick. Attention over learned object embeddings enables complex visual reasoning. *Advances in neural information processing systems*, 34:9112–9124, 2021.

[11] M. Ding, Z. Chen, T. Du, P. Luo, J. Tenenbaum, and C. Gan. Dynamic visual reasoning by learning differentiable physics models from video and language. *Advances in Neural Information Processing Systems*, 34:887–899, 2021.

[12] N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, S. Welleck, P. West, C. Bhagavatula, R. Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] C. Fan, X. Zhang, S. Zhang, W. Wang, C. Zhang, and H. Huang. Heterogeneous memory enhanced multimodal attention model for video question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1999–2007, 2019.

[14] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu. Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 2334–2343, 2017.

[15] D. Fougnie. The relationship between attention and working memory. *New research on short-term memory*, 1:45, 2008.

[16] D. Gao, L. Zhou, L. Ji, L. Zhu, Y. Yang, and M. Z. Shou. Mist: Multi-modal iterative spatial-temporal transformer for long-form video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14773–14783, 2023.

[17] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[18] M. Grunde-McLaughlin, R. Krishna, and M. Agrawala. Agqa: A benchmark for compositional spatio-temporal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11287–11297, 2021.

[19] M. Grunde-McLaughlin, R. Krishna, and M. Agrawala. Agqa 2.0: An updated benchmark for compositional spatio-temporal reasoning. *arXiv preprint arXiv:2204.06105*, 2022.

[20] J. Hsu, J. Mao, J. Tenenbaum, and J. Wu. What's left? concept grounding with logic-enhanced foundation models. *Advances in Neural Information Processing Systems*, 36, 2024.

[21] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[22] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 804–813, 2017.

[23] R. Hu, A. Rohrbach, T. Darrell, and K. Saenko. Language-conditioned graph networks for relational reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10294–10303, 2019.

[24] F. Huang, K. Lu, C. Yuxi, Z. Qin, Y. Fang, G. Tian, and G. Li. Encoding recurrence into transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

[25] D. Hudson and C. D. Manning. Learning by abstraction: The neural state machine. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] D. A. Hudson and C. D. Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018.

[27] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[28] D. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur. Block-recurrent transformers. *Advances in Neural Information Processing Systems*, 35:33248–33261, 2022.

[29] S. Jaiswal, B. Fernando, and C. Tan. Tdam: Top-down attention module for contextually guided feature selection in cnns. In *European Conference on Computer Vision*, pages 259–276. Springer, 2022.

[30] A. Jha, B. Patro, L. Van Gool, and T. Tuytelaars. Barlow constrained optimization for visual question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1084–1093, 2023.

[31] C. Jing, Y. Jia, Y. Wu, X. Liu, and Q. Wu. Maintaining reasoning consistency in compositional visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5099–5108, 2022.

[32] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[33] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision*, pages 2989–2998, 2017.

[34] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[35] W. Kim, B. Son, and I. Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] T. M. Le, V. Le, S. Venkatesh, and T. Tran. Hierarchical conditional relation networks for video question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9972–9981, 2020.

[38] J. Lei, L. Li, L. Zhou, Z. Gan, T. L. Berg, M. Bansal, and J. Liu. Less is more: Clipbert for video-and-language learning via sparse sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7331–7341, 2021.

[39] M. Lewis, N. V. Nayak, P. Yu, Q. Yu, J. Merullo, S. H. Bach, and E. Pavlick. Does clip bind concepts? probing compositionality in large image models. *arXiv preprint arXiv:2212.10537*, 2022.

[40] C. Li, H. Xu, J. Tian, W. Wang, M. Yan, B. Bi, J. Ye, H. Chen, G. Xu, Z. Cao, et al. mplug: Effective and efficient vision-language learning by cross-modal skip-connections. *arXiv preprint arXiv:2205.12005*, 2022.

[41] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[42] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[43] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[44] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, pages 121–137. Springer, 2020.

[45] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[46] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[47] J. Lu, V. Goswami, M. Rohrbach, D. Parikh, and S. Lee. 12-in-1: Multi-task vision and language representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10437–10446, 2020.

[48] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[49] J. Mao, X. Yang, X. Zhang, N. Goodman, and J. Wu. Clevrer-humans: Describing physical and causal events the human way. *Advances in Neural Information Processing Systems*, 35:7755–7768, 2022.

[50] D. Mascharka, P. Tran, R. Soklaski, and A. Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4942–4950, 2018.

[51] D. K. Nguyen, V. Goswami, and X. Chen. Movie: Revisiting modulated convolutions for visual counting and beyond. In *International Conference on Learning Representations*, 2020.

[52] R. OpenAI. Gpt-4 technical report. *ArXiv*, 2303, 2023.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[54] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[55] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[56] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[57] R. Shrestha, K. Kafle, and C. Kanan. Answer them all! toward universal visual question answering models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10472–10481, 2019.

[58] A. Suhr, M. Lewis, J. Yeh, and Y. Artzi. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–223, 2017.

[59] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, 2019.

[60] H. Tan and M. Bansal. Object ordering with bidirectional matchings for visual reasoning. In *Proceedings of NAACL-HLT*, pages 444–451, 2018.

[61] H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, 2019.

[62] A. Urooj, H. Kuehne, B. Wu, K. Chheu, W. Bousselham, C. Gan, N. Lobo, and M. Shah. Learning situation hyper-graphs for video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14879–14889, 2023.

[63] M. Vaishnav and T. Serre. Gamr: A guided attention model for (visual) reasoning. In *The Eleventh International Conference on Learning Representations*, 2022.

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] J. Wang, G. Chen, Y. Huang, L. Wang, and T. Lu. Memory-and-anticipation transformer for online action understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13824–13835, 2023.

[66] J. Wang, Y. Ge, R. Yan, Y. Ge, K. Q. Lin, S. Tsutsui, X. Lin, G. Cai, J. Wu, Y. Shan, et al. All in one: Exploring unified video-language pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6598–6608, 2023.

[67] P. Wang, A. Yang, R. Men, J. Lin, S. Bai, Z. Li, J. Ma, C. Zhou, J. Zhou, and H. Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR, 2022.

[68] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som, et al. Image as a foreign language: Beit pretraining for vision and vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19175–19186, 2023.

[69] Y. Wang, K. Li, Y. Li, Y. He, B. Huang, Z. Zhao, H. Zhang, J. Xu, Y. Liu, Z. Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022.

[70] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[71] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[72] B. Wu, S. Yu, Z. Chen, J. B. Tenenbaum, and C. Gan. Star: A benchmark for situated reasoning in real-world videos. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[73] C.-Y. Wu, Y. Li, K. Mangalam, H. Fan, B. Xiong, J. Malik, and C. Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022.

[74] Q. Wu, Z. Lan, K. Qian, J. Gu, A. Geramifard, and Z. Yu. Memformer: A memory-augmented transformer for sequence modeling. In Y. He, H. Ji, S. Li, Y. Liu, and C.-H. Chang, editors, *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 308–318, Online only, Nov. 2022. Association for Computational Linguistics.

[75] Y. Wu, Y. Zhao, B. Hu, P. Minervini, P. Stenetorp, and S. Riedel. An efficient memory-augmented transformer for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2210.16773*, 2022.

[76] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016.

[77] K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

[78] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

[79] S. Yu, J. Cho, P. Yadav, and M. Bansal. Self-chained image-language model for video localization and question answering. *arXiv preprint arXiv:2305.06988*, 2023.

[80] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6281–6290, 2019.

[81] P. Zhang, X. Li, X. Hu, J. Yang, L. Zhang, L. Wang, Y. Choi, and J. Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5579–5588, 2021.

A Appendix / supplemental material

B Elaborated Experiments and Results Discussion

B.1 Further comparisons on CLEVR-Humans, CLEVR-CoGenT and NLVRv1

Here, we provide further comparisons with benchmark-specific methods for CLEVR-Humans [33], CLEVR-CoGenT [32] and NLVRv1 [58] (not reported in main paper due to space limitations). As mentioned in main paper, these benchmarks utilize synthetic images and are a test of pure visual reasoning capabilities that are minimally influenced by increased world knowledge or usage of stronger visual backbones.

CLEVR-Humans as already mentioned in main paper evaluates a model's reasoning generalization capabilities to unseen scenarios or question forms. CLEVR-CoGenT studies compositional attribute generalization. Specifically, it has two conditions – i) cond.A wherein all cubes have color $\in \{gray, blue, brown, yellow\}$ and cylinders $\in \{red, green, purple, cyan\}$ (spheres can be any color), and ii) cond.B wherein color-sets are switched b/w cubes and cylinders. A model is then trained on one condition and evaluated on both the original and alternate condition. A higher accuracy on the alternate condition indicates that the model learns more 'compositionally' as it generalizes better to novel shape-color combinations with less feature/attribute combination overfitting.

Table 5: Elaborated results on CLEVR-Humans (left), CLEVR-CoGenT (middle) and NLVRv1 (right). IPRM achieves state-of-art across the three benchmarks and does not require additional supervision such as bounding boxes or functional programs. * requires func. programs supervision / pre-defined dataset-specific neural modules. ▼ requires object bounding-boxes supervision.

Model	CLV-Hum	
	ZS	FT
PG+EE* [33]	54.0	66.6
NS-VQA▼* [78]	-	67.8
RAMEN [57]	57.8	-
FiLM [55]	56.6	75.9
GLT [4]	-	75.8
LEFT [20]	-	78.8
MAC [26]	57.4	81.5
MDETR▼ [34]	59.9	81.7
IPRM	**63.8**	**85.5**

Model	CoGenTr-A		CoGenFT-B	
	ValA	ValB	ValA	ValB
NS-VQA▼* [78]	**99.8**	63.9	-	-
MDETR▼ [34]	**99.8**	76.7	-	-
StackAtt-MLP[76]	80.3	68.7	75.7	75.8
PG + EE* [32]	96.6	73.7	76.1	92.7
Tbd-Net* [50]	98.8	75.4	96.9	96.3
MAC [26]	99.0	78.3	97.2	96.1
FILM [55]	98.3	78.8	81.1	96.9
IPRM	99.1	**80.3**	**98.0**	**98.2**

Model	NLVR1
	Test-U
CNN-RNN [58]	56.3
MAC [26]	59.4
FILM [55]	61.2
NMN* [2]	62.0
N2NMN* [22]	66.0
CNN-BiATT [60]	66.1
IPRM (scratch)	63.8
IPRM-CLV-FT	**73.0**

Finally, NLVRv1 evaluates language-grounded visual reasoning. Each sample of this benchmark comprises a set of three synthetic images and a composite natural language statement about the images which can evaluate to True or False and requires various visual-linguistic reasoning skills.

As shown in table 5, IPRM achieves state-of-art results across the three benchmarks and does not require pre-annotated bounding-boxes or functional programs as additional supervision. For **CLEVR-Humans** (table 5 left), it outperforms larger-scale models such as MDETR and RAMEN in zero-shot performance even though the latter is pre-trained on multiple VQA datasets. It also increases state-of-art in finetuned setting by 3.8%.

For **CLEVR-CogenT** (table 5 centre) , IPRM achieves the highest generalization results amongst methods in both the CoGen-Train A and Finetune B. Specifically, it obtains 80.3% acc. on cond. B (when trained on cond. A), which is 1.5% higher than the previous state-of-art cond.B method FILM and 3.6% higher than MDETR. When further finetuned on cond.B, IPRM generalizes for both cond.A and cond.B achieving 98.0% and 98.2% unlike FILM which overfits to cond.B and thereby has poor performance on cond.A. Further, its performance on cond.A (99.1%) is highest amongst methods that do not utilize bounding box or localization supervision and marginally lower than MDETR and NS-VQA (which utilize bounding-box supervision).

Finally, **for NLVRv1** (table 5 right), IPRM model trained from scratch achieves 63.8% acc. and performs competitively with existing task-specific state-of-art model CNN-BiAtt. When finetuned from its CLEVR checkpoint, we find IPRM achieves 73.0% acc. which is 7% higher than existing visual inputs state-of-art for NLVRv1 and suggests strong reasoning transfer capabilities of IPRM. It further outperforms the N2NMN method which requires pre-defined neural modules to be identified for the dataset.

B.2 Elaborated STAR results and ablations for video reasoning tasks

We provide results on the STAR Test, further baselines and model ablations for video reasoning tasks in table 6.

B.3 CLIP Integration Results

We provide results with additional CLIP [56] backbones including CLIP VIT-L/14, CLIP VIT-B/16 and CLIP VIT-L/14@336px on GQA [27], NLVRv2 [59] and CLEVR-Humans in table 7. We compare with alternate prominent vision-language attention mechanisms including Cross-att and Concat-att blocks as well as a simple joint projection of vision and language pooled representations (referred as Wt-Proj-Att). As shown in the table, IPRM can enhance performance for the CLIP

Table 6: **Left:** Results on STAR [72] official hidden test set (evaluation server) with ground-truth vision (GT V) and predicted vision (PR V); **Right:** Results on STAR val. set with num. of sampled frames =32 unless otherwise stated in (); IPRM outperforms prior state-of-art SeviLA-BLIP2 VLM across question types.

Model	Setup	STAR-Test				
		Int.	Seq.	Pred.	Feas.	Avg.
Vis-BERT[43]	GT V	34.7	35.9	31.2	31.4	34.7
CLIP-BERT[38]	GT V	36.3	38.9	30.7	29.8	36.5
NS-SR[72]	GT V	42.6	46.3	43.4	43.9	44.5
IPRM	GT V	**70.5**	**83.8**	**85.3**	**79.1**	**79.7**
Vis-BERT[43]	-	33.6	37.2	31.0	30.8	34.8
CLIP-BERT[38]	-	39.8	43.6	32.2	31.4	36.7
NS-SR[72]	PR V	30.9	31.8	30.2	29.7	30.7
SHG-VQA [62]	-	48.0	42.0	35.3	32.5	39.5
GF [3]	-	56.1	61.3	52.7	45.7	53.9
mPLUG [40]	-	60.4	65.6	57.5	49.6	58.3
IPRM	PR V	**61.7**	**72.7**	**75.4**	**71.3**	**70.3**

Model	Int.	Seq.	Pred.	Feas.	Avg.
All-in-One [66]	47.5	50.8	47.7	44.0	47.5
Temp[ATP](32) [5]	50.6	52.8	49.3	40.6	48.3
MIST [16]	55.5	54.2	54.2	44.4	51.1
InternVideo(8) [69]	62.7	65.6	54.9	51.9	58.7
SeViLA-BLIP2 [79]	63.7	70.4	63.1	62.4	64.9
Concat-Att-2L	58.6	64.8	71.0	66.5	63.5
Concat-Att-4L	60.2	66.9	70.8	64.7	64.9
Cross-Att-4L	60.0	67.2	68.9	68.4	65.0
Concat-Att-6L	59.1	66.4	70.7	65.5	64.4
Cross-Att-6L	52.0	57.6	60.4	55.9	55.4
IPRM(m1,t1)	57.8	65.1	71.0	65.3	63.2
IPRM(m1,t9)	63.1	70.3	76.5	68.9	68.1
IPRM(m6,t1)	62.0	70.2	72.7	68.5	67.5
IPRM(m6,t9)(16)	62.9	70.0	**76.9**	67.1	68.1
IPRM(m6,t9)	**64.2**	**72.9**	75.3	**69.1**	**69.9**

Table 7: **Left:** Comparison of IPRM with prominent vision-language attention mechanisms with CLIP VIT-L/14 backbones on CLEVR-Humans, GQA and NLVRv2 benchmarks ('4L' indicates 4 att layers; 'x' indicates model did not converge). **Right:** Results with other CLIP variants VIT-B and VIT-L@ 336 on GQA and NLVRv2. Refer suppl. sec B.3 for further discussion.

Model (CLIP VIT-L/14 bbone)	+Param	+GFLOPs	GQA TestD	NLVR2 Test	CLV-H ZS	CLV-H FT
Wt-Proj-Fusion	0.6M	0.1	53.5	60.8	58.5	74.4
Cross-Att (2L)	9.2M	1.5	55.1	62.1	-	-
Concat-Att (2L)	7.2M	4.4	55.3	60.5	-	-
Cross-Att (4L)	17.6M	3.1	57.4	54.4	60.3	80.0
Concat-Att (4L)	13.6M	8.9	58.1	55.9	61.2	81.1
Cross-Att (6L)	26.0M	4.5	56.8	x	60.8	80.4
Concat-Att (6L)	19.7M	13.3	57.4	x	62.0	81.8
IPRM	5.2M	5.9	**59.3**	**65.1**	**64.3**	**84.6**

Model (CLIP VIT-B/16 bbone)	GQA TestD	NLVR2 Test
Wt-Proj-Fusion	51.4	59.9
Cross-Att	54.6	56.6
Concat-Att	56.0	57.4
IPRM	55.9	60.8

Model (CLIP VIT-L/14@336)	GQA TestD	NLVR2 Test
Wt-Proj-Fusion	54.0	61.1
Cross-Att	57.4	58.4
Concat-Att	57.3	59.1
IPRM	59.4	65.4

variants across GQA, NLVRv2 and CLV-Humans in comparison to concat and cross-att blocks. Further, it is more parameter efficient with only 5.5M additional parameters in comparison to 4-layer as well as 2-layer stacks of Cross-Att (9.2M 2-layer, 17.6M 4-layer) and Concat-Att (7.2M 2-layer, 13.6M 4-layer). With regards to computational FLOPs, IPRM consumes 5.9GFLOPs which is marginally higher than Cross-Att 4-layer config (3.1GFLOPs) and lower than Concat-Att 4-layer config (8.9GFLOPs). Note, that the performance benefits of adding further layers of cross- or concat-att blocks are observed to be minimal after 4 layers, and can also depend on the amount of training data available. E.g. Both cross- and concat-att blocks of 2 layers had better performances on NLVRv2 (which has a limited set of training questions relative to GQA and CLEVR) in comparison to 4 layer config.

B.4 Further reasoning computation visualizations

We provide elaborate reasoning computation visualizations of IPRM showing the lang. and vis. attentions across parallel operations and computation steps during *operation formation* and *operation execution* stages. Fig. 8 shows a scenario wherein IPRM correctly utilizes parallel and iterative computations to compute intermediate operations of "find object close to front", "retrieve/compare shape and size", "find applicable objects with both same shape and size". Fig. 9 shows another correct prediction of IPRM, and this time, its intermediate reasoning visualization is useful to determine that the entailed reasoning appears sensible. Fig. 10 shows an incorrect prediction by IPRM and its intermediate reasoning visualizations also suggest that IPRM did not understand the question and thereby did not attend to relevant objects. Finally, Fig. 11 shows a scenario wherein while IPRM produces the correct answer, it's intermediate reasoning appears imprecise which makes the prediction (and underlying reasoning) less reliable. We provide further visualizations with a CLIP

VIT-L/14 backbone on GQA samples in the supplemental jupyter notebook output (html format for easier viewing).

C Model implementation and experiment details

We implement IPRM in PyTorch [53] as a generic vision-language module receiving a set of input vision (or scene-representation) tokens and input language (or task-representation) tokens. We provide **Python-style pseudocode of IPRM in figs 12, 13 and 14**. For all experiments, we set the internal dimension of IPRM to 512 and use the same configuration of num. parallel operations (N_{op})=6, num. computation steps (T)=9, reduction ratio (r)=2 and window size (W)=2. We follow benchmark-specific conventions for vision-language backbones that are detailed below in sec. C.1. For CLIP [56], we utilize the official models from Huggingface [70]. All experiments are performed on a single NVIDIA A40 GPU with 46GB memory and averaged over 3 trials with different random seeds wherever possible (done for primary experiments on STAR, AGQA, CLEVRER-Humans, CLEVR-Humans). Unless otherwise specified, the learning rate is initialized to 1e-4 with Adam [36] optimizer and gradient clipping value of 8. The learning-rate is reduced based on validation acc. plateau with reduction factor 0.5, threshold 0.001 and patience 0. Further experiment hyper-parameters and settings are provided below. **Source code for experiments and visualization along with model checkpoints will be released publicly via Github**.

C.1 Benchmark-specific experiment details

CLEVR-Humans. We use the CLEVR-Humans dataset from [33] which comprises images from original CLEVR dataset [32] and human crowdsourced questions. We use a batch size of 216 for training. We use the same language encoder (Distil-Roberta[46] from Huggingface[71]) as in existing state-of-art MDETR [34] and frozen ResNet101 backbone layer 3 spatial features (as in [26, 50, 33]). We perform all ablation experiments with 14x14x1024 visual features. Each ablation model is first pretrained for 10 epochs on the original CLEVR dataset (the initial learning rate for IPRM is 1e-4 and for language encoder is 1e-5) and then finetuned on CLEVR-Humans for 40 epochs with early stopping (learning rate of 1e-4 throughout). As observed in prior work [50], we similarly found in multiple scenarios with occluded objects that visual attention only partially identified such objects. Hence, we simply resampled (bilinear sampling) visual input to obtain 16x16x1024 features and empirically found more complete visual attentions with a corresponding 1.1% improvement in accuracy. The final two best performing model configurations (*Nop=6, T=9, W=2, R=2* and *Nop=6, T=9, W=2, R=1*) from ablations were then pre-trained for 35 epochs on CLEVR and finetuned on CLEVR-Humans. While we found that configuration *Nop=6, T=9, W=2, R=1* obtains highest zero-shot (ZS) acc. of 65.6% and finetuned (FT) acc. of 86.3%, we adopt *Nop=6, T=9, W=2, R=2* (with 63.3% ZS and 85.4% FT acc.) as our optimal model given its lesser parameters and FLOPs.

GQA. We use the GQA compositional real-world image question answering dataset from [27]. Based on prior VQA methods on GQA [27, 23, 43, 31], we utilize pre-extracted bounding-box object proposal features and object label predictions obtained from a pretrained object detector [17, 81]. The bounding box coordinates is normalized to range of 0 to 1 based on the original input image size, and the 4 coordinates are transformed to a distributed representation through a learned nonlinear projection. This representation is concatenated with a learned projection of the predicted object labels (initialized with glove[54] 300dim embeddings) to form the final visual input. We train IPRM for 25 epochs with a batch-size of 192 and same hyperparameters as before. We evaluate the final model on both the test-dev split and the official test evaluation server / hidden test set (`https://eval.ai/featured-challenges/225/evaluation`). Our test eval server submission is anonymous with only submission id and method name used ('#7024-IPRM') and a randomly generated team name ('sn12').

STAR-VideoQA. We use the STAR-VideoQA dataset for situational reasoning on real-world videos from [72]. Based on previous videoQA methods [72, 38, 43] for STAR, we utilize object bounding boxes, labels, human pose and human-object relations across frames (note: we do not use the situation hyper-graphs or functional programs). We first perform experiments with the provided ground truth object bounding boxes, labels and human-object relations as well as provided human pose predictions from Alphapose [14] as reported in main paper. Each of these is projected to a distributed representation through learned non-linear projections to obtain object token-wise representations.

A further learnable positional embedding for each frame is added to these representations which are then flattened across frames to form the visual input to IPRM. For the language encoder, we found both a simple Bi-LSTM and Distil-roberta language encoder obtain similar performance, and hence choose the simpler Bi-LSTM as the language model. We evaluate models on both 16 uniformly sampled frames and 32 uniformly sampled frames, and empirically found using 32 frames has $\sim$0.9% higher performance. Since reported models in [72] use 16 frames, we report on the same setting in main paper. A batch size of 64 was used with learning rate 1e-4 over 30 epochs with early stopping. We evaluated the models on both the validation split and official test evaluation server `https://eval.ai/web/challenges/challenge-page/1325/overview`. Our submission is anonymous with only submission id and method name used (*'#9644-IPRM'*) and a randomly generated team name (*'sn12'*). For the all-predicted (no ground-truth) visual input setup, similar to [72], we utilize a fasterRCNN [17] object extractor, ST-Trans scene graph extractor [8] and the same Alphapose predictor to obtain predicted object bounding boxes, labels, human-object relations and human poses. We observe a drop of $\sim$11% in predicted setup similar to observations in [72], suggesting further performance can be achieved through better object and relationship detection backbones.

AGQAv2. We use the AGQAv2 [18, 19] benchmark that comprises balanced training and test splits. We followed the same methodology as in STAR. however, since AGQAv2 comprises a larger number of questions and increased diversity in language, we used a distil-roberta language encoder instead of a bi-LSTM.

CLEVRER-Humans. We use the CLEVRER-Humans dataset introduced in [49] for temporal, physical and causal video reasoning which comprises videos from the original CLEVRER dataset [77]. Similar to in STAR-VideoQA and neurosymbolic models [77, 78], we utilize a pretrained faster-RCNN based object localization and attribute prediction network from [77]. We again form object-level representations by concatenating learned projections of object-bounding box coordinates and predicted object attributes (i.e. color, shape and material). A frame-level learnable positional embedding is added and object-tokens across frames are flattened to form the final visual input to IPRM. For the language encoder, we used a simple bi-LSTM similar to existing methods. Note we do not use the functional programs or event causal graphs in our model. The batch size was 128 with a learning rate of 8e-5 and every 4 frames sampled (resulting on average 32 sampled frames). We evaluated models in the three setups – from scratch, zero-shot (CLEVRER-pretrained) and finetuned (CLEVRER-pretrained). Since the CLEVRER-Humans dataset is relatively small (comprising only 1076 questions $\sim$ 8 batches; $\sim$ 0.5% of original CLEVRER), for scratch training we trained for 250 epochs (with early stopping) while for finetuning we finetuned for 150 epochs (with 35 epochs for original CLEVRER training).

CLEVR-CoGen. We use the CLEVR-CoGen dataset from [32] and follow the same setup as in CLEVR-Humans. We use a simpler bi-LSTM language encoder for experiments since the questions are synthetic program-generated unlike in CLEVR-Humans (crowdsourced free-form). We trained our model on condition A for 40 epochs (with early stopping) and used the best cond. A validation performance model to evaluate generalization performance on cond.B. For finetuning on cond.B we finetuned the best cond.A model for 20 epochs and used the best cond.B validation performance model to also evaluate on cond.A. All other hyperparameters are the same as mentioned for CLEVR-Humans.

NLVR. We use the NLVRv1 and NLVRv2 datasets from [58, 59]. NLVRv1 comprises 3 synthetic images and a language statement while NLVRv2 comprises 2 real-world images and a lang. statement. For both datasets, the obtained visual tokens for each images was flattened to obtain the final visual input and an image-wise positional embedding was added to indicate image order. For the language encoder, we used a simple Bi-LSTM.

D Limitations

Our work proposes a new "iterative" and "parallel" reasoning mechanism (IPRM) designed to address complex visual reasoning and question answering (VQA) scenarios. While we studied IPRM through experiments across various VQA benchmarks along with quantitative ablations and qualitative reasoning visualizations, we note some possible limitations of IPRM in this section. Similar to existing VQA and deep-learning methods, IPRM may reflect biases that are present in the training distribution of VQA benchmarks. This may lead it to overfit to certain image inputs or question

forms and possibly provide skewed answers in such scenarios. Further, the utilized vision-language backbones in our experiments may also entail visual, language and cultural biases in their original training distribution which may permeate to IPRM upon integration for VQA scenarios. In this regard, we hope the capability to visualize intermediate reasoning of IPRM and diagnose its error cases (as discussed in main paper Sec. 4.4) can serve a useful tool to benefit interpretability in VQA and identify possible reasoning biases that may emerge in the model.

E Potential Negative Impact

In relation to VQA and deep-learning methods in general, the deployment of IPRM in real-world applications without thorough consideration of dataset or training distribution biases, could inadvertently reinforce existing vision, language and cultural biases present in the data, leading to erroneous outcomes or skewed answers. Further, the deployment of VQA methods such as IPRM in sensitive domains such as healthcare or scene/footage analysis could raise ethical concerns, including privacy violations, algorithmic reliability, and the potential for unintended consequences stemming from erroneous or biased predictions.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and introduction reflects the motivation of method and experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Limitations are provided in appendix.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Benchmark and training details are provided along with module pseudocode and example CLIP integration. Source code for experiments will made publicly available along with checkpoints.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer:[Yes]

   Justification: Paper provides training and implementations details and provides module pseudocode. Full source code will be made publicly available upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix provides details on hyperparameters and dataset specific settings for all experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix mentions that results were averaged over atleast 3 seeds for primary experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix mentions the type of GPU and its memory for all experiments along with batch size of experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Authors reviewed code of ethics during submission.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer:[Yes]

    Justification: Paper mentions potential negative impacts of work in appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: answerNA

Justification: Paper does not explicitly pose such risks; however in limitations and potential negative impact this is mentioned.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Yes, all coding libraries and datasets are properly cited and credited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:No new asset

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Paper does not involve crowdsourcing or human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Paper does not involve crowdsourcing or human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

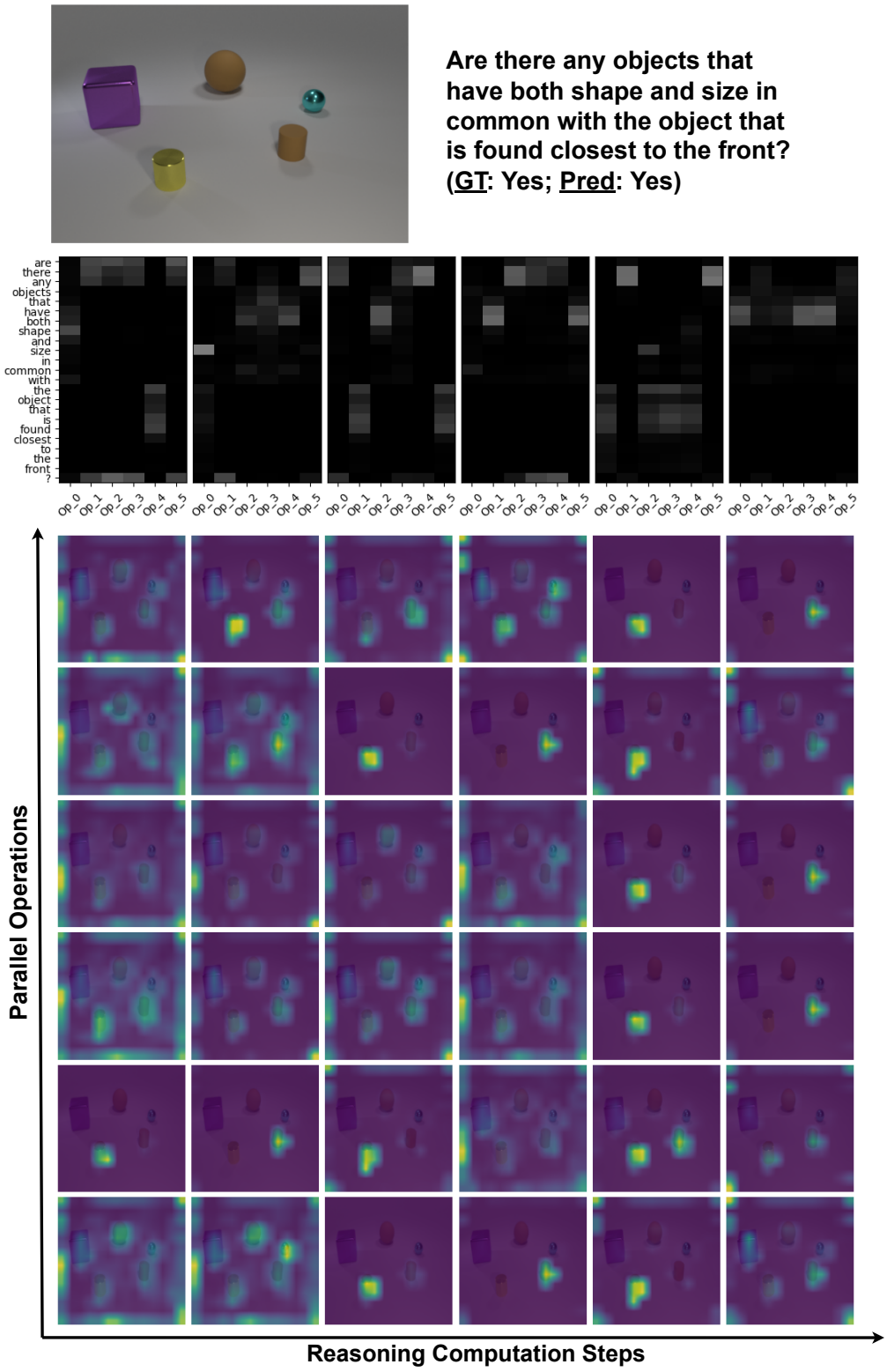

Figure 8: **Top**: original image and question; **middle**: language attentions across parallel operations (clubbed together; op_k represents parallel operation k) and computation steps. **Bottom**: Visual attentions across parallel ops and computation steps. Here, IPRM correctly utilizes parallel and iterative compute to locate the correct candidate object for prediction (to which all operations attend in last step).

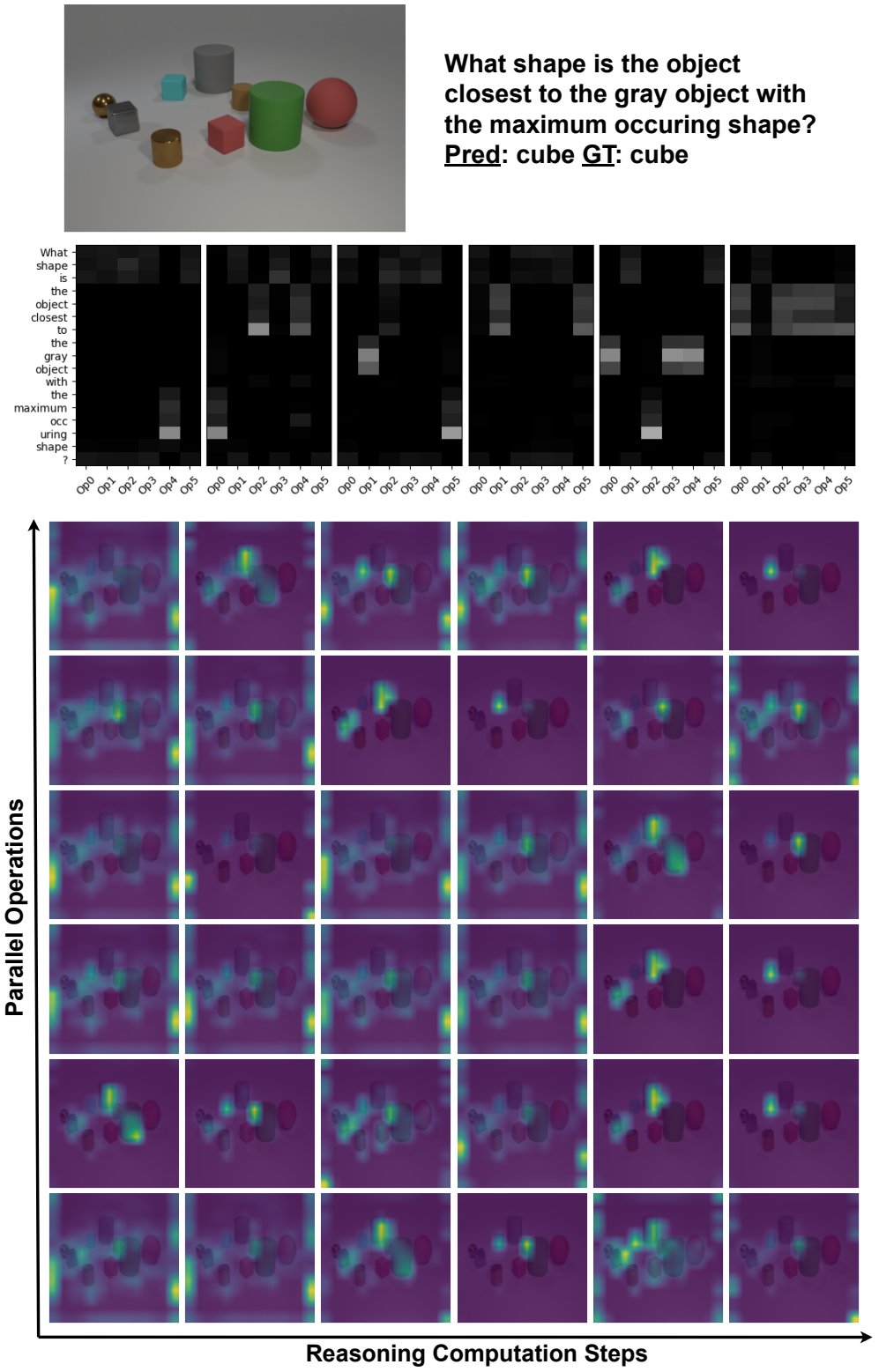

Figure 9: In this example, IPRM predicts the correct answer and its visual attention trace provides evidence of correct intermediate reasoning. In penultimate reasoning step, IPRM correctly localizes the gray object with maximum occuring shape (cylinder) and in the final step, the parallel operations attend to both the cyan cube and the brown cylinder closest to previously identified gray cylinder.

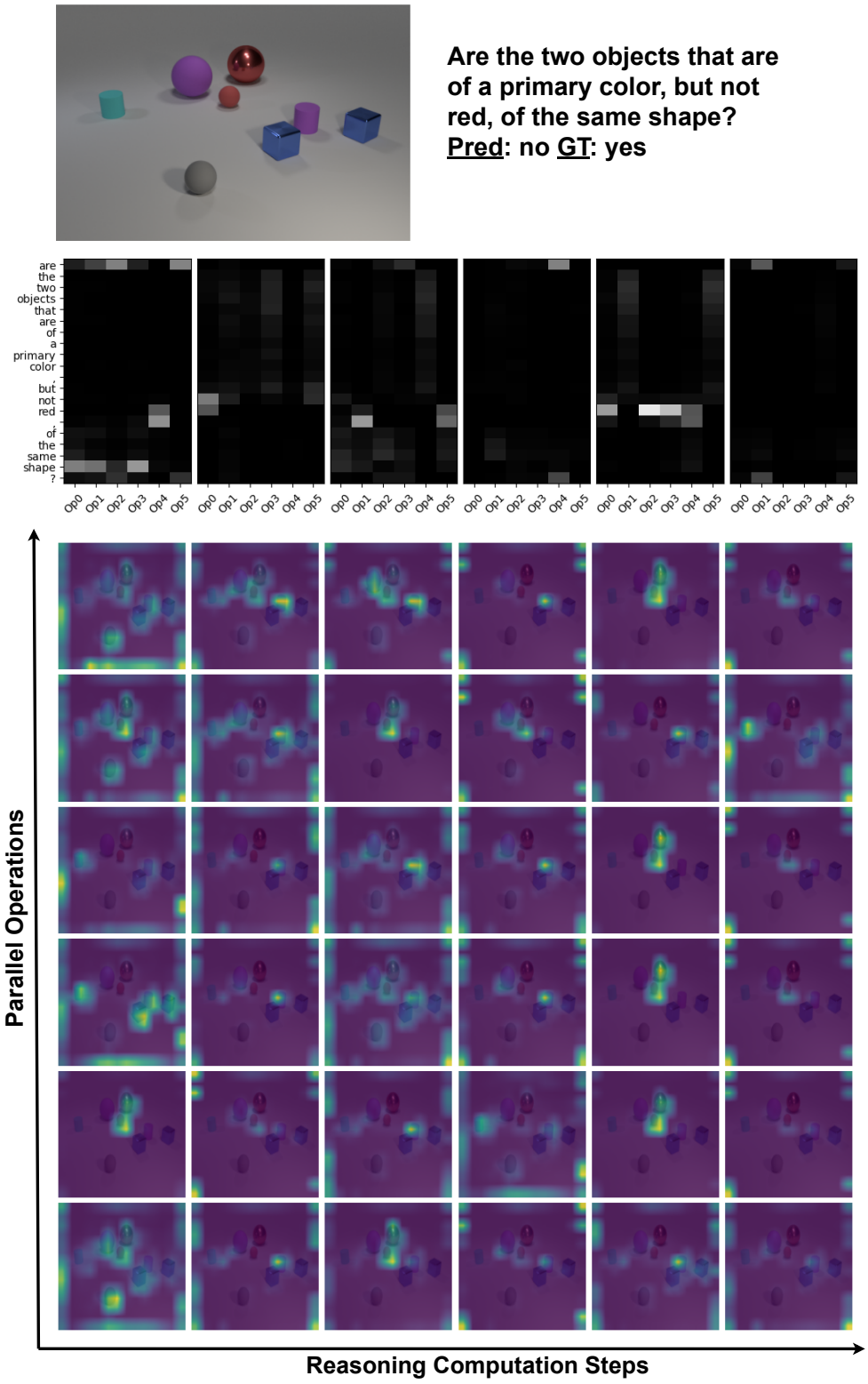

Figure 10: Example where IPRM outputs incorrect answer and the intermediate reasoning appears faulty possibly due to lack of understanding what a “primary color is". The pair of blue (a primary color) cubes in this case should have been identified but are not visually attended in any of the operations across reasoning steps).

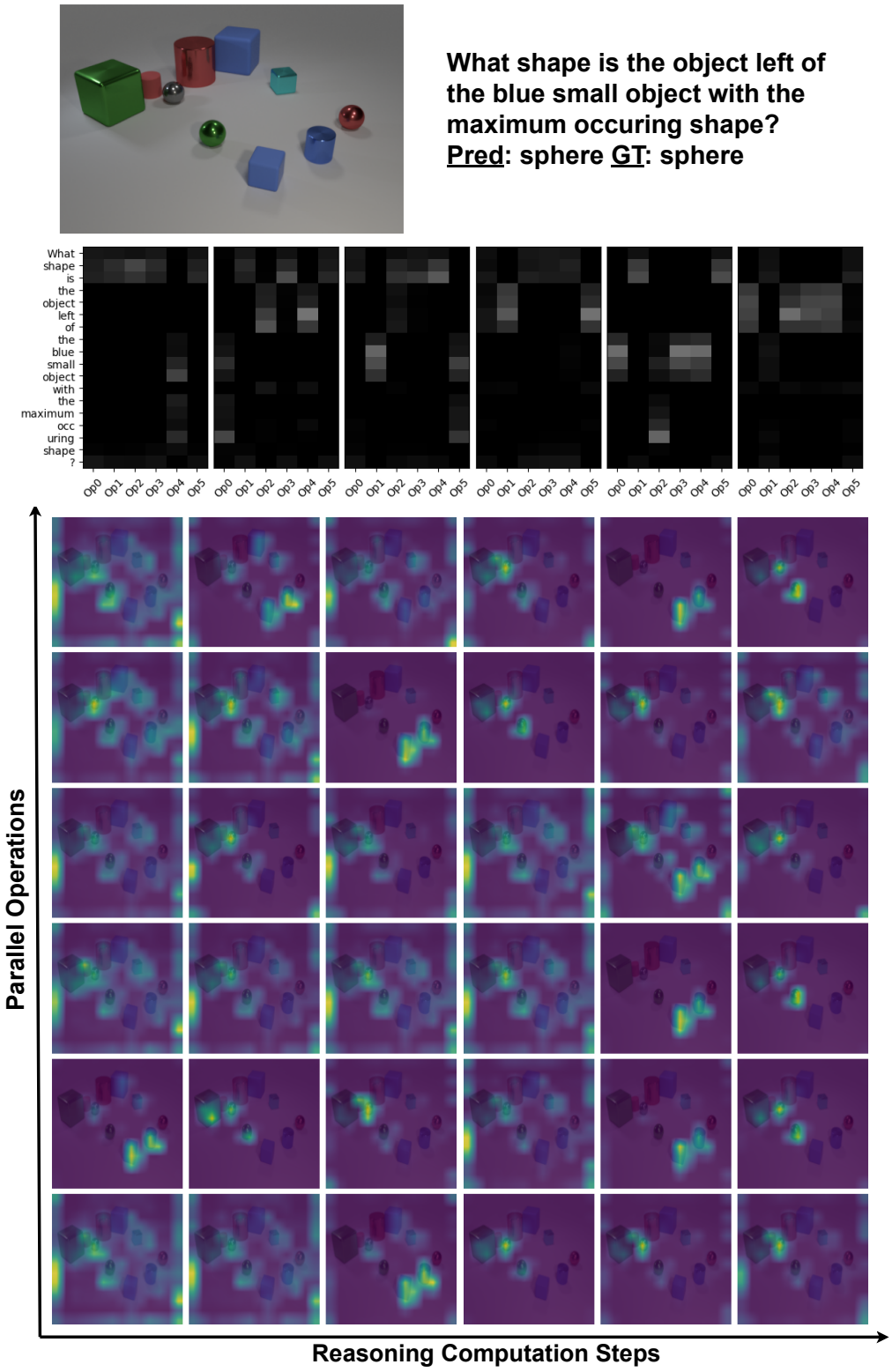

Figure 11: Example wherein IPRM produces correct answer but its visual attention trace suggests intermediate reasoning may be imprecise. The maximum occuring shape is cube; however both the blue small cylinder and blue small cube appear to be attended in the penultimate step as the “blue small object with max occuring shape" making the reasoning and prediction less reliable.

```
def iprm_forward(vis_tokens, #B×Nv×Dm
                 lang_tokens, #B×Nl×Dm
                 lang_summary_rep, #B×Dm
                 num_parallel_ops=6,
                 num_iterative_steps=9,
                 mem_window_len=2):
    mem_op_states = []
    mem_res_states = []
    lang_atts = []
    vis_atts = []

    #0. Initialize memory
    b, d = vis_tokens.size(0), vis_tokens.size(-1)
    mem_op_state, mem_res_state = _init_mem_state(num_parallel_ops, b,d)
    mem_op_states.append(mem_op_state)
    mem_res_states.append(mem_res_state)
    for i in range(num_iterative_steps):
        #1. Form new set of latent operations from lang. token features
        new_ops, lang_att = operation_formation(lang_tokens, mem_op_state)

        #2. Execute new operations on vis. input to form new results
        new_ops_results, vis_att = operation_execution(vis_tokens, new_ops,
        ↪  mem_res_state)

        #3. Apply operation composition
        mem_op_state, mem_res_state = operation_composition(new_ops,
        ↪  new_ops_results, mem_op_states, mem_res_states)

        #4. Maintain memory states within lookback window
        mem_op_states.append(mem_op_state)
        mem_res_states.append(mem_res_state)
        mem_op_states = mem_op_states[min(-1, -mem_window_len):]
        mem_res_states = mem_res_states[min(-1, -mem_window_len):]

        #5. Store lang. and vis. atts for visualization
        lang_atts.append(lang_att)
        vis_atts.append(vis_att)

    #6. 'Pool' final result
    final_result = pool_final_result(mem_res_state, mem_op_state,
    ↪  lang_summary_rep)

    return final_result, lang_atts, vis_atts
```

Figure 12: IPRM pseudocode (1/3)

```
#Below, "Lin" refers to a linear layer
#and `"MLP" refers to a 2-layer multi-layer-perceptron layer
def operation_formation(lang_tokens, #B×Nl×Dm
                        prev_op_state #B×Nop×Dm (Nop=num parallel ops)
                        ):
    #1. Form new op "query" based on prior op state
    op_q = MLP_l(prev_op_state) #paper eq. 4

    #2. Use lang_token_feats as attn "key" and "value" (paper eq. 5)
    lang_k = lang_tokens
    lang_v = lang_tokens

    #3. Retrieve new latent ops from lang. rep through attention
    latent_ops, lang_attn = mod_attn(op_q, lang_k, lang_v,
                                     lang_attn_proj) #paper eq.6; L194

    return latent_ops, lang_attn

def operation_execution(vis_tokens, #B×Nv×Dm
                        new_ops, #B×Nop×Dm
                        prev_res_state): #B×Nop×Dm
    #1. Form feature modulation weights (paper eq.7)
    s_v = concat([Lin_op(new_ops), Lin_res(prev_res_state)]) #concat across feat.
    ↪  axis
    s_v = Lin_s(s_v)

    #2. Form visual attention "key" (paper eqs. 8 and 9)
    vis_red_rep =  Lin_v1(vis_tokens)
    mod_vis = s_v * vis_red_rep
    Nop = mod_vis.size(1)
    vis_k = MLP_v(concat([mod_vis, vis_red_rep])) #concat across feat. axis

    #3. Form visual attention "query" and "value" (paper eq. 10)
    vis_q = Lin_op_q(new_ops)
    vis_v = Lin_v2(vis_tokens)

    #4. Obtain new latent "results" through vis attention (paper eq.11)
    latent_results, vis_attn = mod_attn(vis_q, vis_k, vis_v, vis_att_proj)

    return latent_results, vis_attn

def mod_attn(q, k, v, att_proj_layer, attn_mask):
    qk_mult = q*k  #element-wise product
    attn_wt = att_proj_layer(qk_mult) #linear projection (paper L194)
    attn_wt = softmax(attn_wt + (attn_mask * -1e30))
    out = (attn_wt * v).sum() #sum across feature axis
    return out, attn_wt
```

Figure 13: IPRM pseudocode (2/3)

```
def operation_composition(new_ops, #B×Nop×Dm
                          new_res, #B×Nop×Dm
                          mem_op_states, #list of W elements: B×Nop×Dm
                          mem_res_states #list of W elements: B×Nop×Dm
                          ):
    #1. Integrate new-ops and results into memory (paper eq. 12 and 13)
    inter_op_state = Lin_op_u(new_ops) + Lin_op_h(mem_op_states[-1])
    inter_res_state= Lin_res_u(new_res) + Lin_res_h(mem_res_states[-1])

    #2. Concat operation and result states over memory lookback window
    op_states_windowed = concat([inter_op_state, mem_op_states])
    res_states_windowed = concat([inter_res_state, mem_res_states])

    #3. Form inter-operation queries and keys (paper eq. 14)
    op_queries = Lin_op_q(inter_op_state)
    op_keys = Lin_op_k(op_states_windowed)

    #4. Form inter-operation op values and res values (paper eq. 15-16)
    op_values = Lin_op_v(op_states_windowed)
    res_values = Lin_res_v(res_states_windowed)

    #5. Compute inter-operation attention (paper eq. 17)
    attn_mask = identity_matrix(op_keys.size(1))[:op_queries.size(1)]
    new_op_state, op_attn_wt = mod_attn(op_queries, op_keys, op_values,
    ↪  op_attn_proj, attn_mask)

    #6. Obtain new operation and result states (paper eq. 18-19)
    new_op_state = new_op_state + Lin_op_u2(inter_op_state)
    new_res_state = op_attn_wt*new_res_state + Lin_res_v2(inter_res_state)

    return new_op_state, new_res_state

def _init_mem_state(num_parallel_ops, b):
    #slice specified num parallel ops from initialized params ~ N(0,1)
    op_init_state = op_init_param[:num_parallel_ops]
    res_init_state= res_init_param[:num_parallel_ops]
    #broadcast batch-wise to get B×N_op×Dm
    return op_init_state.repeat(b,1,1), res_init_state.repeat(b,1,1)

def pool_final_result(res_state, op_state, lang_summary_rep):
    pool_q = Lin_pq(lang_summary_rep)
    pool_k = Lin_pk(op_state)
    return mod_attn(pool_q, pool_k, res_state)
```

Figure 14: IPRM pseudocode (3/3)