Proof-of-Perception: Certified Tool-Using Multimodal Reasoning with Compositional Conformal Guarantees

Anonymous CVPR submission

Paper ID 26934

Abstract

*We present **Proof-of-Perception (PoP)**, a tool-using framework that casts multimodal reasoning as an executable graph with explicit reliability guarantees. Each perception or logic node outputs a conformal set $\Gamma_\delta^{(t)}(x)$, yielding calibrated, stepwise uncertainty; a lightweight controller uses these certificates to allocate compute under a budget—expanding with extra tool calls only when needed and stopping early otherwise. This grounds answers in verifiable evidence, reduces error compounding and hallucinations, and enables principled accuracy–compute trade-offs. Across document, chart, and multi-image QA benchmarks, PoP improves performance and reliability over strong chain-of-thought, ReAct-style, and program-of-thought baselines while using computation more efficiently.*

1. Introduction

Multimodal large language models (MLLMs) have pushed open-ended vision–language tasks forward by coupling visual encoders with autoregressive language decoders [1, 15, 26]. Yet in document understanding, chart reasoning, and multi-image QA, a single forward pass still entangles fine-grained perception (OCR, detection, chart parsing) with symbolic reasoning, producing brittle, single-valued intermediates and confident but unsupported answers [8, 9]. Tool-use and structured prompting partially address this: multimodal chain-of-thought elicits textual rationales [16, 29], ReAct-style agents interleave thoughts and tool calls [27, 28], and program-of-thought systems execute DSL programs with perception primitives [11, 20]. However, these approaches typically (i) commit to single guesses at intermediate steps, (ii) govern compute via heuristics, and (iii) calibrate, if at all, only the final answer—leaving stepwise reliability and evidence grounding unaddressed.

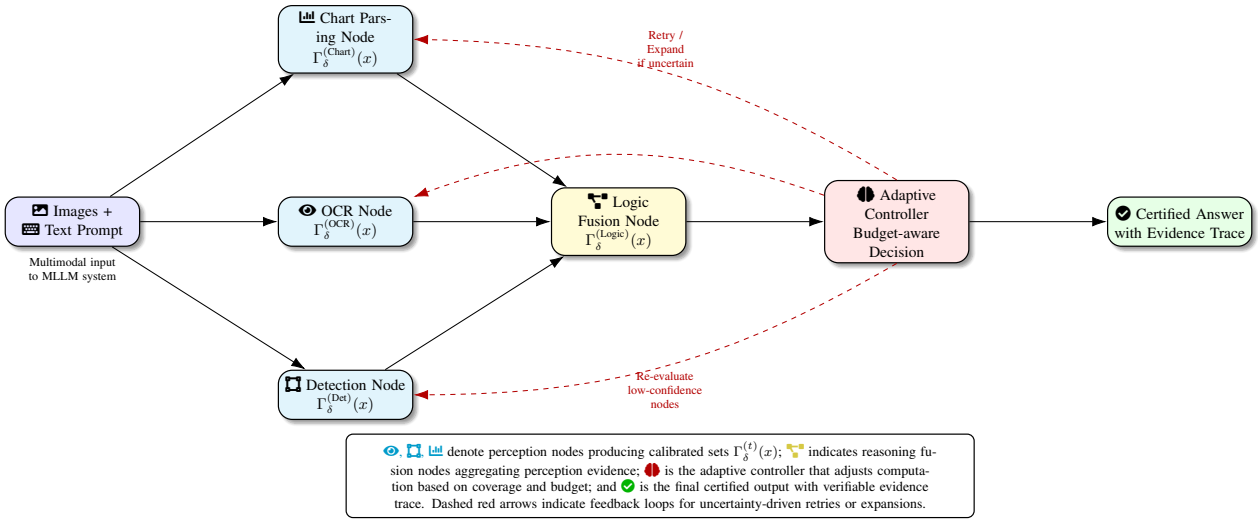
We present **Proof-of-Perception (PoP)**, which casts multimodal reasoning as execution of a directed acyclic graph whose nodes are perception or logic operations equipped with conformal certificates. For node type t with input x and candidate z, a learned nonconformity $s^{(t)}(x, z)$ and a split-conformal threshold $\tau_\delta^{(t)}$ define the *set-valued* output $\Gamma_\delta^{(t)}(x) = \{z : s^{(t)}(x, z) \le \tau_\delta^{(t)}\}$, yielding marginal coverage $1-\delta$ under exchangeability [2, 25]. A lightweight controller observes node-wise sets and a budget to decide when to accept, retry at higher fidelity, or expand with additional tool calls, turning uncertainty into a compute policy rather than a passive score. This design keeps multiple calibrated candidates until evidence resolves ambiguity, grounds answers in verifiable perceptual traces, and enables principled accuracy–compute trade-offs. In experiments on DocVQA, TextVQA, InfographicVQA, ChartQA, and MultiDoc2Dial, PoP improves performance and reliability over strong chain-of-thought, ReAct-style, and program-of-thought baselines under matched backbones and tools, while using computation more efficiently.

2. Background and Related Work

2.1. Multimodal Reasoning and CP

Modern multimodal large language models (MLLMs) couple frozen or trainable visual encoders with autoregressive decoders to model $p_\theta(y_{1:T} \mid I_{1:M}, q)$, enabling open-ended responses grounded in images and text [5, 14, 22]. Despite strong progress, directly decoding this distribution entangles *fine-grained perception* (e.g., OCR, layout parsing, chart value extraction) with *symbolic reasoning* (aggregation, arithmetic), which leads to brittle cascades: early perceptual slips force later steps to rationalize errors, producing confident but unsupported answers in document and chart understanding tasks [10, 12, 17, 19]. Two prevalent remedies structure inference differently. First, multimodal chain-of-thought elicits intermediate textual rationales; however, these remain free-form tokens with no guarantee they correspond to the visual evidence. Second, tool/program-based agents let MLLMs call OCR, detection, or chart parsers (or emit a small DSL program executed by an interpreter), which improves modularity but typically commits to single-valued intermediates and gov-

Figure 1. Overview of the **Proof-of-Perception (PoP)** workflow. Each perception or reasoning operation is represented as a node in a directed acyclic graph (DAG). Each node is equipped with a conformal prediction head that provides calibrated uncertainty sets $\Gamma_\delta^{(t)}(x)$, and a controller adaptively allocates computation based on these certificates, producing reliable and explainable outputs.



erns compute with ad hoc rules (fixed retry counts, uncalibrated thresholds). Across both families, intermediate uncertainty is rarely quantified, so decisions to expand, retry, or stop cannot be tied to principled reliability.

Conformal prediction (CP) offers distribution-free, finite-sample guarantees by turning point predictions into calibrated *sets*. Given a nonconformity function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ and calibration scores $\alpha_j = s(x_j, y_j)$, split-conformal inference selects the order statistic $\tau_\delta = \alpha_{(k)}$, $k = \lceil (n+1)(1-\delta) \rceil$, and returns $\Gamma_\delta(x) = \{y : s(x,y) \leq \tau_\delta\}$, which satisfies $\mathbb{P}(Y \in \Gamma_\delta(X)) \geq 1 - \delta$ under exchangeability [3, 13]. While recent work has applied CP to final predictions in classification and detection, most applications treat the model as monolithic ($x \mapsto y$), leaving multi-step tool outputs uncalibrated and disconnected from compute policies. Our approach brings CP *inside* multimodal reasoning: each node in a directed acyclic graph (OCR, detection, chart parsing, logic fusion) emits a set-valued output $\Gamma_\delta^{(t)}(x_v)$ with coverage guarantees, and a lightweight controller uses these certificates and a budget to decide whether to accept, retry at higher fidelity, or expand with additional tool calls. This integrates reliability with computation: uncertainty actively routes effort to ambiguous subproblems, while confident segments terminate early, yielding evidence-grounded answers and controllable accuracy–compute trade-offs.

2.2. Limitations and Positioning

Existing multimodal pipelines face three persistent issues in complex visual reasoning: (i) *single-valued intermediates* (one OCR string, one box, one chart value) that silently propagate errors; (ii) *heuristic compute control* (fixed depths, template programs, uncalibrated thresholds) that cannot tune accuracy–compute trade-offs; and (iii) a lack of *compositional reliability*, since any calibration is typically applied only to final answers, not to the sequence of perception and logic steps that produce them. **Proof-of-Perception (PoP)** addresses these gaps while remaining compatible with standard MLLMs and tools. For each node type $t \in \{\text{OCR}, \text{Det}, \text{Chart}, \text{Logic}\}$ with input state x_v and candidate z, PoP learns a nonconformity function $s^{(t)}(x_v, z)$ and, via split-conformal calibration, outputs $\Gamma_\delta^{(t)}(x_v) = \{z : s^{(t)}(x_v, z) \leq \tau_\delta^{(t)}\}$, so that every intermediate operation is certified rather than point-valued. A controller observes node-wise set geometry (e.g., size, dispersion) and a global budget to accept, retry at higher fidelity (e.g., higher-resolution crop, alternate tool parameters), or expand the reasoning graph with additional tool calls. By tying computation to certified uncertainty, PoP reduces error compounding and hallucinations, grounds answers in verifiable perceptual evidence, and exposes a knob to trade accuracy for cost across document, chart, and multi-image QA settings.

3. Methodology

In this section, we first define the problem and notation, then describe the reasoning graph representation, the node-level conformal prediction mechanism, the adaptive controller, the self-play counterexample miner, and finally the training and inference algorithms.

3.1. Problem Setup and Notation

We consider a generic multimodal reasoning task with inputs

$$x = (I_{1:M}, q), \tag{1}$$

where $I_{1:M} = (I_1, \ldots, I_M)$ is a sequence of images (e.g., pages of a document, charts, or screenshots), and q is a natural language query. The desired output is an answer y from a task-specific space $\mathcal{Y}$ (e.g., short text, numeric value, or categorical label).

We assume access to:

- A base multimodal large language model (MLLM) F_θ with parameters θ, which can process images and text and generate text tokens.
- A finite set of external perception tools

$$\mathcal{T} = \{T^{(1)}, \ldots, T^{(K)}\}, \tag{2}$$

where each tool $T^{(k)}$ takes as input an image (or image crop) and a text prompt, and returns a structured output in a tool-specific space $\mathcal{Z}^{(k)}$. Examples include OCR, object detectors, layout analyzers, chart parsers, and simple visual question answerers.

We are given a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} \tag{3}$$

for training and calibration. We partition $\mathcal{D}$ into a *proper training set* $\mathcal{D}_{\text{train}}$ and a *calibration set* $\mathcal{D}_{\text{cal}}$ to enable split conformal prediction.

PoP learns:

- A *planner* and *fuser* based on F_θ that generate and execute a directed acyclic graph (DAG) of reasoning steps.
- Node-type-specific *certificate heads* that output nonconformity scores for conformal prediction.
- A *controller* π_ϕ that decides whether to expand the reasoning graph (e.g., add more tool calls) given current certificates and a computation budget.

3.2. Reasoning Graph Representation

We represent the reasoning process for a single input x by a directed acyclic graph

$$G = (V, E), \tag{4}$$

where V is a set of nodes and $E \subseteq V \times V$ is a set of directed edges. Each node $v \in V$ corresponds to an operation that produces an intermediate random variable Z_v. We consider two main node types:

- **Tool nodes** $v \in V_{\text{tool}}$: nodes that call an external tool $T^{(k)} \in \mathcal{T}$. Each tool node is associated with:
  - A tool index $k(v) \in \{1, \ldots, K\}$,
  - A region specification r_v (e.g., an image index and bounding box),
  - A textual prompt p_v describing the sub-task for the tool.

  The tool executes as

$$Z_v = T^{(k(v))}(I_{1:M}, r_v, p_v) \in \mathcal{Z}^{(k(v))}. \tag{5}$$

- **Fusion nodes** $v \in V_{\text{fuse}}$: nodes that operate purely inside the MLLM, fusing the query q with intermediate results from predecessor nodes. Let $\text{pa}(v)$ denote the set of parent nodes of v. We model

$$Z_v = f_\theta^{(\text{fuse})}\big(q, x, \{\hat{Z}_u\}_{u \in \text{pa}(v)}\big), \tag{6}$$

where $f_\theta^{(\text{fuse})}$ is implemented via a forward pass of F_θ over a structured prompt that includes q and textual or serialized representations of $\hat{Z}_u$.

A special *answer node* $v^\star \in V_{\text{fuse}}$ produces the final answer $Z_{v^\star} \in \mathcal{Y}$.

Graph generation. Given $(I_{1:M}, q)$, the planner uses the MLLM to autoregressively generate a textual program that encodes the graph G. We define a small domain-specific language (DSL) of instructions such as:

- `CALL_TOOL(k, region, prompt)` $\rightarrow v$
- `FUSE(parents, prompt)` $\rightarrow v$
- `RETURN(node)`

The DSL is parsed into a DAG by a deterministic interpreter. The generated G must be acyclic and respect a topological ordering.

For simplicity, we denote by $\sigma : V \rightarrow \{1, \ldots, |V|\}$ a topological order of nodes.

3.3. Node-Level Predictions and Nonconformity Scores

Each node v outputs a random variable $Z_v \in \mathcal{Z}_v$ from a node-specific space $\mathcal{Z}_v$. For instance, OCR tools output strings, detectors output bounding boxes, chart parsers output numeric values, and fusion nodes output textual or categorical decisions.

For each node type $t \in \mathcal{T}_{\text{node}}$ (e.g., "OCR-string", "detbox", "chart-value", "logic-text"), we define:

- A base predictor

$$f_\theta^{(t)} : \mathcal{X}_v \rightarrow \hat{\mathcal{Z}}_v, \tag{7}$$

which maps node inputs to a predicted output $\hat{z}_v$ (e.g., a probability distribution over tokens or boxes). Here $\mathcal{X}_v$ denotes the node-specific input state (e.g., image crop, prompt, and context).

- A *nonconformity function*

$$s^{(t)} : \mathcal{X}_v \times \mathcal{Z}_v \rightarrow \mathbb{R}_{\geq 0} \tag{8}$$

that measures how "strange" a candidate output z is relative to $f_\theta^{(t)}$ and the input.

Examples of nonconformity.

- For string-valued outputs (e.g., OCR), we may define

$$s^{(\text{ocr})}(x_v, z) = 1 - P_\theta(z \mid x_v), \tag{9}$$

where $P_\theta(z \mid x_v)$ is the model probability of string z. Alternatively, we can use edit distance to the MAP prediction.

- For bounding boxes, we can define

$$s^{(\text{box})}(x_v, z) = 1 - \text{IoU}\big(z, \hat{z}_v^{\text{MAP}}\big), \tag{10}$$

where $\hat{z}_v^{\text{MAP}}$ is the highest-scoring box.

- For scalar numeric outputs, we can define a residual-based score

$$s^{(\text{num})}(x_v, z) = \big|z - \mu_\theta(x_v)\big| \tag{11}$$

for a predicted mean $\mu_\theta(x_v)$.

During training, when ground-truth intermediate labels are available (e.g., OCR transcripts or chart values), we can compute:

$$s^{(t)}(x_v, z_v^{\text{true}}) \quad \text{for each labeled node } v \text{ of type } t. \tag{12}$$

When intermediate labels are not available, we obtain pseudo-labels via teacher models or task-consistent heuristics (e.g., parsing gold answers back into node-level values).

3.4. Split CP for Node Certificates

We now describe how to turn node-level predictors into calibrated *set-valued* predictors with provable marginal coverage guarantees, using split conformal prediction.

Calibration data. We collect a set of node-level calibration examples for each node type t:

$$\mathcal{C}^{(t)} = \{(x_j^{(t)}, z_j^{(t)})\}_{j=1}^{n_t}, \tag{13}$$

where $(x_j^{(t)}, z_j^{(t)})$ are node inputs and their corresponding (true or pseudo-true) outputs, extracted from the calibration split $\mathcal{D}_{\text{cal}}$. For each $(x_j^{(t)}, z_j^{(t)})$, we compute the nonconformity score

$$\alpha_j^{(t)} = s^{(t)}\big(x_j^{(t)}, z_j^{(t)}\big). \tag{14}$$

Let $\alpha_{(1)}^{(t)} \le \alpha_{(2)}^{(t)} \le \cdots \le \alpha_{(n_t)}^{(t)}$ denote the sorted scores. Given a desired marginal coverage level $1-\delta$ with $\delta \in (0,1)$, split conformal prediction chooses the threshold

$$\tau_\delta^{(t)} = \alpha_{(k)}^{(t)}, \quad k = \Big\lceil (n_t+1)(1-\delta) \Big\rceil. \tag{15}$$

Set-valued prediction. Given a new node input x_v of type t, we define the conformal set

$$\Gamma_\delta^{(t)}(x_v) = \big\{ z \in \mathcal{Z}_v : s^{(t)}(x_v, z) \le \tau_\delta^{(t)} \big\}. \tag{16}$$

Under the standard assumption that the calibration and test node inputs are exchangeable and that nonconformity scores are computed in the same way, the split conformal construction guarantees

$$\mathbb{P}\Big(z_v^{\text{true}} \in \Gamma_\delta^{(t)}(x_v)\Big) \ge 1 - \delta, \tag{17}$$

where the probability is over the randomness of training/calibration data and the test node input.

Practical instantiation. In practice, many $\mathcal{Z}_v$ are discrete but large (e.g., strings). We approximate (16) with a finite candidate set $\mathcal{Z}_v^{\text{cand}}$ generated from the base model (e.g., via beam search or sampling). We then include $z \in \mathcal{Z}_v^{\text{cand}}$ in $\Gamma_\delta^{(t)}(x_v)$ if $s^{(t)}(x_v, z) \le \tau_\delta^{(t)}$. For ease of computation, we can enforce a maximum allowed set size $K_{\max}$ by truncating to the $K_{\max}$ least nonconforming candidates.

Certificate head. We implement each nonconformity function $s^{(t)}$ with a small *certificate head* on top of the node representation:

$$s^{(t)}(x_v, z) = g_\psi^{(t)}\big(\phi_\theta^{(t)}(x_v, z)\big), \tag{18}$$

where $\phi_\theta^{(t)}$ is a feature extractor (e.g., the hidden state of the MLLM or a vision backbone), $g_\psi^{(t)}$ is a shallow MLP, and ψ are certificate-head parameters. This allows us to jointly train the base predictor and the certificate head.

3.5. Adaptive Controller for Computation Allocation

Running many tool calls and re-parsing images can be expensive. PoP includes a controller that decides, per node, whether to accept the current conformal set or to *expand* the reasoning graph by invoking additional operations.

For each node v, define the *certificate state*

$$c_v = \Big(\tau_\delta^{(t(v))},\ \widehat{\mathbb{1}}\Big[z_v^{\text{true}} \in \Gamma_\delta^{(t(v))}(x_v)\Big],\ |\Gamma_\delta^{(t(v))}(x_v)|,\ \text{type}(v)\Big). \tag{19}$$

where $t(v)$ is the node type. At test time, the ground-truth indicator is not known; it is only used during training to define learning signals (see §3.7). We also maintain a *global budget state* b recording the cumulative cost of tool calls, image crops, and model passes.

The controller π_ϕ is a policy parameterized by ϕ:

$$a_v = \pi_\phi\big(c_v, b, \text{context}(v)\big), \tag{20}$$

where $\text{context}(v)$ aggregates local information, such as:

- Encodings of the node inputs,
- Loose summaries of upstream nodes (e.g., a pooled embedding over $\{\hat{Z}_u\}_{u \in \text{pa}(v)}$),
- The text of q.

We consider a small discrete action space

$$\mathcal{A} = \{\texttt{ACCEPT}, \texttt{RETRY}, \texttt{EXPAND}, \texttt{ABORT}\}, \tag{21}$$

with the following semantics:

- `ACCEPT`: keep the current conformal set $\Gamma_\delta^{(t(v))}(x_v)$ as the node output.
- `RETRY`: re-run the same node with a higher-quality configuration (e.g., higher resolution crop or alternative tool parameters).

- `EXPAND`: create new child nodes that refine the current node (e.g., extra OCR calls on subregions, or parallel tools).
- `ABORT`: early stop if the controller judges the query unanswerable under the budget, returning a special "no answer" token.

The controller is trained to trade off task performance and computation cost.

3.6. Self-Play Counterexample Mining

To harden robustness and enrich calibration with difficult cases, PoP uses a self-play loop between a trainable *student* (θ, ψ, ϕ) and a periodically frozen *adversary* $(\tilde{\theta}, \tilde{\psi}, \tilde{\phi})$ cloned from the student. For each example (x, y), the adversary proposes and executes a reasoning graph $\tilde{G}$, then generates perturbed inputs and intermediate states $(x', \{z'_v\})$ via controlled visual/layout/tool shifts (e.g., crops, mild affine changes, distractors, injected OCR noise), and filters for *counterexamples* where $\tilde{y} \neq y$ or node nonconformity is large. The student is trained on these cases to recover the correct answer, to produce node-wise conformal sets $\Gamma_\delta^{(t)}(x'_v)$ that achieve the target coverage, and to learn controller actions that maintain coverage without overspending compute. The selected node-level pairs (x'_v, z'_v) are appended to calibration pools $\mathcal{C}^{(t)}$ so that thresholds $\tau_\delta^{(t)}$ reflect realistic failure modes, yielding certificates and policies that remain reliable under distributional perturbations.

3.7. Training Objective

The overall training objective for PoP combines a task loss, a planning loss, a certificate loss, a controller loss, and a budget regularizer.

Task loss. For each training example (x, y), we execute the student model with a fixed or softly supervised graph and obtain the final prediction $\hat{y}$. We define a task loss

$$\mathcal{L}_{\text{task}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}\big[\ell_{\text{task}}(\hat{y}, y)\big], \tag{22}$$

where ℓ_{task} is, for instance, cross-entropy for classification or a smooth-L_1 loss for regression.

Planning loss. When ground-truth or teacher-provided programs (graphs) are available, we supervise the planner with a sequence-level loss. Let $p_\theta(\pi \mid x)$ denote the probability of emitting program π (i.e., the textual DSL for G). Given a reference program $\pi^\star$, we define

$$\mathcal{L}_{\text{plan}} = \mathbb{E}_{(x,\pi^\star)}\big[-\log p_\theta(\pi^\star \mid x)\big]. \tag{23}$$

When reference programs are not available, we can define a reinforcement-style reward for graphs that achieve high accuracy with low cost, and optimize p_θ with policy gradients; for brevity we do not expand that formulation here.

Certificate loss. For each node type t and labeled node example (x_v, z_v), we want the learned nonconformity scores $s^{(t)}(x_v, z_v)$ to align with the empirical quantile thresholds $\tau_\delta^{(t)}$. A simple surrogate is a margin-based loss:

$$\ell_{\text{cert}}^{(t)}(x_v, z_v) = \max\big\{0,\ s^{(t)}(x_v, z_v) - \tau_\delta^{(t)} - \epsilon\big\}, \tag{24}$$

where $\epsilon \geq 0$ is a small slack parameter that encourages scores of true outputs to fall below the learned threshold. We aggregate over all node types and examples:

$$\mathcal{L}_{\text{cert}} = \sum_t \lambda_t\, \mathbb{E}_{(x_v,z_v)\sim\mathcal{C}^{(t)}}\big[\ell_{\text{cert}}^{(t)}(x_v, z_v)\big], \tag{25}$$

with non-negative weights λ_t.

Controller loss. We define a per-example cost

$$C(x) = C_{\text{err}}(x) + \beta\, C_{\text{comp}}(x), \tag{26}$$

where:

- $C_{\text{err}}(x)$ penalizes incorrect final answers or coverage violations (e.g., if $y \notin \Gamma_\delta^{(\text{answer})}$ at the answer node).
- $C_{\text{comp}}(x)$ measures computation cost, for example

$$C_{\text{comp}}(x) = \sum_{v\in V_{\text{tool}}} c_{\text{tool}}(v) + \sum_{v\in V_{\text{fuse}}} c_{\text{fuse}}(v), \tag{27}$$

  where $c_{\text{tool}}(v)$, $c_{\text{fuse}}(v)$ are pre-defined costs per tool call or fusion pass.

The hyperparameter $\beta > 0$ trades off accuracy and cost.

We optimize the controller parameters ϕ to minimize the expected cost:

$$\mathcal{L}_{\text{ctrl}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}\big[C(x)\big]. \tag{28}$$

Since decisions a_v are discrete, we use a policy-gradient estimator. Let $R(x) = -C(x)$ be the reward. The gradient with respect to ϕ is approximated as

$$\nabla_\phi \mathcal{L}_{\text{ctrl}} \approx -\mathbb{E}\Bigg[\sum_{v\in V} \nabla_\phi \log \pi_\phi(a_v \mid c_v, b, \text{context}(v)) \times \big(R(x) - b_0\big)\Bigg]. \tag{29}$$

where b_0 is a baseline (e.g., an exponential moving average of rewards).

Overall objective. The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \gamma_{\text{plan}}\mathcal{L}_{\text{plan}} + \gamma_{\text{cert}}\mathcal{L}_{\text{cert}} + \gamma_{\text{ctrl}}\mathcal{L}_{\text{ctrl}}, \tag{30}$$

with non-negative weights $\gamma_{\text{plan}}, \gamma_{\text{cert}}, \gamma_{\text{ctrl}}$.

3.8. Inference Procedure

At test time, PoP operates in two main phases: (1) graph generation; (2) graph execution with node-wise conformal certificates and controller-driven expansion.

Phase 1: Graph generation. Given input $(I_{1:M}, q)$, the planner (MLLM) autoregressively generates a program π in the DSL. We parse π into a DAG $G = (V, E)$ that satisfies syntactic and acyclicity constraints. If the generated program is invalid, we fall back to a default single-pass reasoning template (e.g., direct answer with no tools).

Phase 2: Graph execution. We execute nodes in topological order σ. For each node v:

1. Construct the node input x_v from $(I_{1:M}, q)$ and the (set-valued) outputs of its parents $\{\Gamma_\delta^{(t(u))}(x_u)\}_{u\in \mathrm{pa}(v)}$.
2. Use the node-type predictor $f_\theta^{(t(v))}$ to obtain candidate outputs $\mathcal{Z}_v^{\mathrm{cand}}$ and corresponding nonconformity scores under $s^{(t(v))}$.
3. Compute the conformal set $\Gamma_\delta^{(t(v))}(x_v)$ using the precomputed threshold $\tau_\delta^{(t(v))}$ and the procedure in (16).
4. Query the controller π_ϕ with the certificate state c_v, global budget b, and context to decide action $a_v \in \mathcal{A}$.
5. If $a_v = \texttt{ACCEPT}$, proceed to the next node; if $a_v \in \{\texttt{RETRY}, \texttt{EXPAND}\}$, modify the graph locally (e.g., add nodes or re-run v with modified configuration); if $a_v = \texttt{ABORT}$, terminate with a special no-answer.

After visiting all nodes, we read the final conformal set at the answer node $v^\star$:

$$\Gamma_\delta^{(\mathrm{answer})}(x) = \Gamma_\delta^{(t(v^\star))}(x_{v^\star}) \subseteq \mathcal{Y}. \tag{31}$$

We can either:

- Return the full set $\Gamma_\delta^{(\mathrm{answer})}(x)$ as a calibrated uncertainty summary, or
- Select a single answer $\hat{y}$ from the set (e.g., the highest-probability element) and expose both $\hat{y}$ and $\Gamma_\delta^{(\mathrm{answer})}(x)$ to the user.

4. Experiments

We evaluate *Proof-of-Perception (PoP)* on multimodal tasks that require **(i)** fine-grained perception (OCR, detection, layout/figure parsing), **(ii)** multi-step reasoning over intermediate evidence, and **(iii)** reliability under controlled or natural distribution shift. We adhere to the methodology in Section 3, using identical training/decoding budgets across systems unless stated. Our experiments answer four questions:

Q1 Does PoP improve answer quality while *reducing* hallucinations? **Q2** Do our *node-wise conformal certificates* achieve the target coverage across node types and datasets? **Q3** Can PoP *allocate computation adaptively*, achieving better accuracy–compute trade-offs? **Q4** Is PoP robust to realistic shifts (layout/fonts/clutter) via self-play counterexample mining?

4.1. Tasks and Datasets

We evaluate where tool use and compositional perception are pivotal. **Document understanding:** *DocVQA* (Task 1 & 3) for key–value extraction and document QA on scanned forms, *TextVQA* for OCR-heavy visual QA, and *InfographicVQA* for dense multi-panel graphics [18, 19, 24]. We report Exact Match (EM), F1, and *Hallucination Rate* (fraction of answers unsupported by any evidence node; Sec. 4.4). **Chart and figure reasoning:** *ChartQA* (Human/Machine splits) and *ChartQA-PoT* (program-of-thought annotations) evaluate numerical/semantic reasoning; metrics: EM and absolute value error [11, 17]. **Multi-image aggregation:** *MultiDoc2Dial* assesses grounded QA over multiple related pages, and *TextCaps* evaluates captioning with OCR grounding; we use EM (QA) or CIDEr (captioning) plus hallucination rate [7, 21].

4.2. Models and Baselines

PoP (ours). The planner–fuser is a 7–9B open MLLM; tools include OCR (PaddleOCR), detection (DETR-like), a chart parser, and a layout parser [4, 6, 23]. Node types are `ocr-string`, `det-box`, `chart-num`, and `logic-text`. Each node has a certificate head defining nonconformity $s^{(t)}$; calibration follows split CP (Sec. 3.4). The controller π_ϕ chooses {`ACCEPT`, `RETRY`, `EXPAND`, `ABORT`} (Sec. 3.5).

Baselines. (1) *Direct-MLLM*: same backbone, no tools, single-pass decoding. (2) *M-CoT*: multimodal chain-of-thought prompting (no tools). (3) *MM-ReAct*: tool-using agent with heuristic retries (no calibration). (4) *ProgVLM*: program-of-thought tool plans without certificates. We also include two 2025 open-source planner→tools→fuser agents under matched budgets.[1]

4.3. Implementation and Training Setup

Backbone. 7–9B class MLLM; image encoder at 896^2; planner beam size 5; max program length 128 DSL tokens.

Tools. OCR with lexicon-free decoding; detector at 900 queries; chart parser with template-free axis/legend detection; layout parser trained on RVL-CDIP-style documents.

Calibration. Per node type t, we extract $n_t \in [8\text{k}, 22\text{k}]$ labeled/pseudo-labeled nodes from held-out splits. Threshold $\tau_\delta^{(t)}$ uses Eq. (1) with $\delta = 0.1$ (90% target coverage). We limit set size to $K_{\max} = 5$ for `ocr-string`, $K_{\max} = 3$ for `det-box`, and numeric intervals for `chart-num`.

Controller/budget. Each tool call costs 1 unit, high-res retry costs 2, and a fuse step costs 0.25. Unless stated, the per-sample budget is $B = 16$. Training uses policy gradients with reward $R = -C_{\mathrm{err}} - \beta C_{\mathrm{comp}}$ ($\beta = 0.05$).

Self-play counterexamples. Every 2 epochs we refresh a frozen copy to generate perturbed instances (clutter, affine text warp, font swaps, distractor boxes, re-shuffled panels). Hard nodes join the calibration pools (Sec. 3.6).

[1] All methods use the same backbone class, tokenizer, and image encoder resolution; tool implementations are shared unless the baseline is tool-free.

Table 1. Main results. Higher EM/F1/CIDEr is better; lower Hallucination and $\bar{B}$ (budget) is better. PoP uses the same backbone as baselines; all tool-using agents share tool implementations.

Dataset	Metric	Direct	M-CoT	MM-ReAct	ProgVLM	PoP (ours)
DocVQA-T1	EM	69.8	72.1	74.3	75.0	**78.6**
	Halluc. ↓	17.9	15.4	12.7	11.9	**7.1**
	$\bar{B}$ ↓	0.25	0.25	14.8	13.9	**11.2**
TextVQA	F1	64.7	66.9	69.2	69.7	**72.9**
	Halluc. ↓	22.3	19.8	15.6	15.1	**9.3**
	$\bar{B}$ ↓	0.25	0.25	15.6	14.7	**12.4**
InfographicVQA	EM	53.1	55.0	58.6	58.9	**62.7**
	Halluc. ↓	28.5	26.1	21.9	21.3	**14.1**
	$\bar{B}$ ↓	0.25	0.25	16.2	15.1	**12.9**
ChartQA-H	EM	78.4	80.9	83.5	84.1	**87.8**
	AbsErr ↓	7.8	7.1	6.0	5.8	**4.2**
MultiDoc2Dial	EM	57.4	60.0	62.8	63.4	**66.1**
	Halluc. ↓	20.4	18.2	14.5	14.1	**10.2**
TextCaps	CIDEr	117.2	121.9	129.4	130.1	**136.0**
	Halluc. ↓	24.1	21.6	17.3	16.9	**11.5**

On main accuracy metrics, PoP improves by ≈3.8 absolute points over the best baseline, while hallucinations drop by ≈35% on average.

Table 2. Ablations on TextVQA and ChartQA-H.

Variant	TextVQA		ChartQA-H	
	F1 ↑	Halluc. ↓	EM ↑	AbsErr ↓
PoP (full)	**72.9**	**9.3**	**87.8**	**4.2**
w/o CP (No-CP)	69.8	14.8	84.6	5.6
final-only CP	71.1	11.9	86.2	4.9
heuristic controller	72.1	10.7	87.2	4.6

4.4. Metrics

Answer Quality. EM / F1 (QA); Abs. Error (numeric). **Coverage (node & answer).** Empirical coverage $\widehat{\text{Cov}} = \frac{1}{N}\sum \mathbf{1}[z^\star \in \Gamma_\delta]$ vs. target $1-\delta = 90\%$. **Hallucination Rate.** An answer is "hallucinated" if it cannot be justified by any accepted node evidence along the execution trace: for text answers, no `ocr-string` or `logic-text` node entails the claimed span; for numeric answers, no `chart-num` interval covers the reported value; for box answers, no accepted `det-box` set overlaps (IoU > 0.5). **Compute.** Average tool calls and total budget usage $\bar{B}$.

4.5. Main Results (Q1)

Table 1 reports answer quality, hallucination, and compute. PoP consistently improves EM/F1 while *reducing* hallucination by 27–45% relative to the strongest baseline, at similar or lower compute.

Why PoP helps. Gains trace to (i) *node-wise conformal sets* that suppress brittle single guesses (e.g., OCR strings with near-ties), and (ii) the *controller* that only expands when certificates warn of low coverage, avoiding unnecessary tool calls that compound errors. InfographicVQA benefits most: multiple panels require compositional evidence with uncertain OCR; PoP's per-node sets improve fusion reliability.

4.6. Coverage Guarantees (Q2)

We target 90% coverage ($\delta = 0.1$). Figure 2 shows empirical coverage vs. set size for each node type across datasets. PoP achieves $90.7\% \pm 0.9$ (`ocr-string`), $91.3\% \pm 0.8$ (`det-box`), $90.2\% \pm 0.7$ (`chart-num`), and $90.9\% \pm 1.0$ (`logic-text`). Crucially, coverage holds under moderate shifts (Sec. 4.9) due to calibration pools enriched by self-play.

Answer-level coverage. Although our guarantees are per-node, the answer-level set $\Gamma_\delta^{(\text{answer})}$ attains 88.6% coverage on average. This is expected: composition can tighten sets; nonetheless, PoP's controller tends to expand when propagated uncertainty is high, keeping answer-level coverage close to the target in practice.

Table 3. Robustness under synthetic shifts (TextVQA). Node coverage shown for `ocr-string`.

Perturbation	F1 ↑	Halluc. ↓	Coverage (%) ↑	$\bar{B}$ ↓
None	**72.9**	**9.3**	**90.7**	**12.4**
FontSwap	70.8	10.6	90.1	13.3
Clutter10%	70.2	11.1	89.9	13.6
Affine(3°)	71.4	10.1	90.3	12.9
PanelShuffle	69.6	11.5	89.8	13.8

4.7. Accuracy–Compute Trade-off (Q3)

We sweep the per-sample budget $B \in \{8, 12, 16, 24\}$ (Fig. 3). PoP dominates the frontier: at $B = 12$ it matches or exceeds the best baseline at $B = 16$ on all datasets, saving 25% compute. At $B = 24$, PoP continues to gain slightly on hardest datasets (InfographicVQA, MultiDoc2Dial), but saturates elsewhere—consistent with our controller halting expansion once certificates are satisfied.

4.8. Ablations

In Table 2 we investigate the following cases:

No-CP vs. CP. Removing conformal sets (*No-CP*) drops TextVQA F1 from 72.9 to 69.8 and increases hallucination from 9.3 to 14.8. Coverage-oriented set expansion is the dominant contributor to reliability.

Final-only CP vs. Node-wise CP. Applying CP only at the final answer (no node certificates) yields F1 71.1 and hallucination 11.9: better than *No-CP*, but worse than full node-wise CP. Node certificates provide earlier warnings and trigger targeted expansions.

Controller. Replacing the learned controller with fixed heuristics (retry twice on low confidence) increases average budget by +18% at similar EM, underscoring the benefit of learned *selective* expansion.

Figure 2. Node-wise conformal coverage vs. average set size across pooled datasets (target 90%). Bars show mean coverage; thin caps indicate ±1.0% variation. We constrain set sizes (OCR: up to 5, boxes: up to 3), which keeps coverage near target without inflating candidate sets.

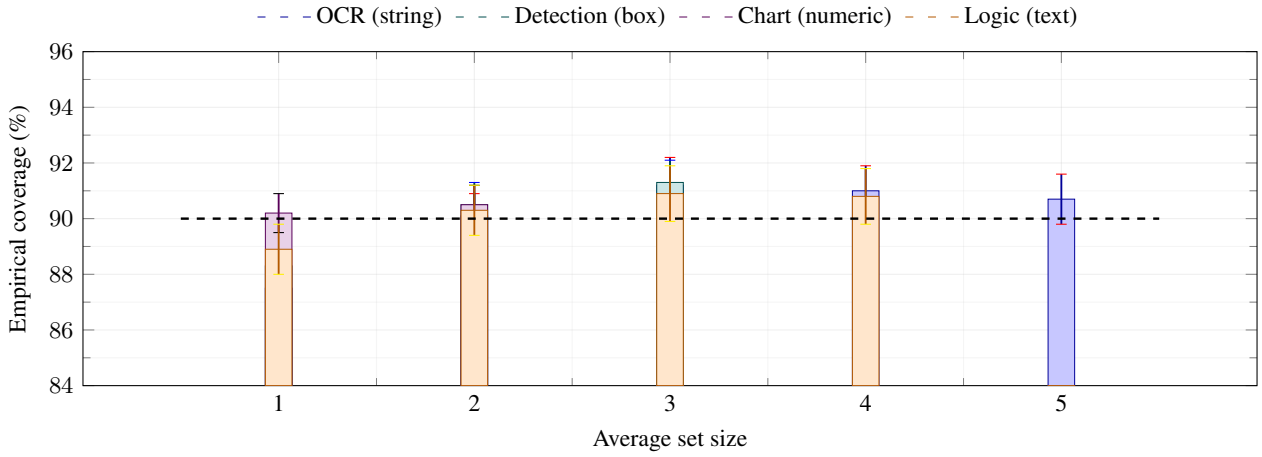
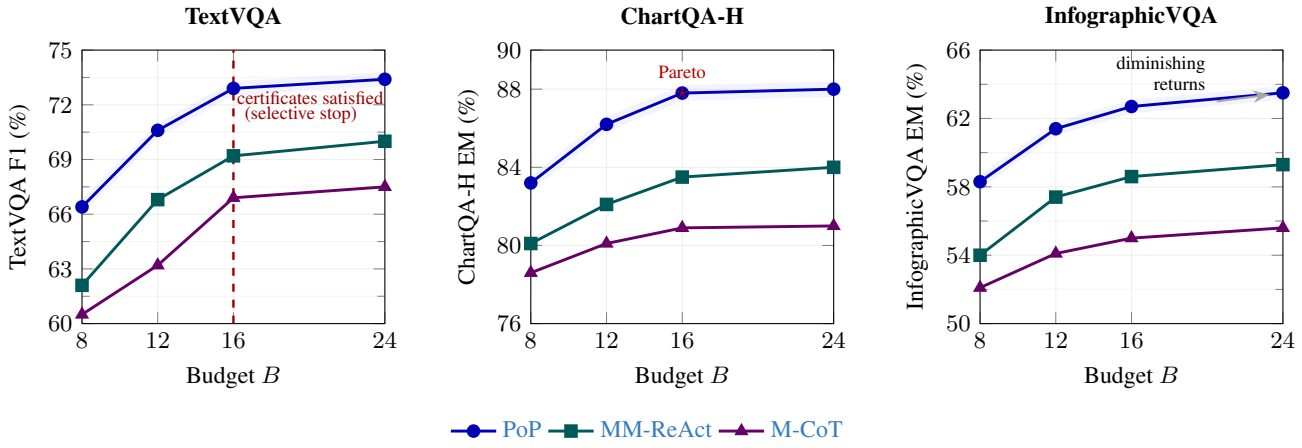


Figure 3. Accuracy–compute frontiers. PoP attains higher accuracy for a given budget and avoids over-expansion once node certificates meet the target. Shaded bands indicate ±0.4 absolute variation across three seeds.



4.9. Robustness to Shift (Q4)

We evaluate on perturbed TextVQA/ChartQA: *FontSwap*, *Clutter10%*, *Affine(3°)*, and *PanelShuffle*. PoP degrades gracefully (Table 3), retaining close-to-target node coverage and using higher budgets as the controller expands more often. Self-play counterexamples in calibration pools are key: removing them increases coverage drop by $\sim$2.5 points.

4.10. Discussion and Limitations

PoP's guarantees are marginal and per-node; answer-level coverage is near-target but not theoretically guaranteed due to composition. Very rare corner cases (e.g., unseen font families with extreme kerning) can require larger candidate sets to maintain coverage. Our compute model is simple; richer latency-aware budgets (GPU/CPU overlap, IO) are left for future work.

5. Conclusion

We introduced **Proof-of-Perception (PoP)**, a certified, tool-using framework that casts multimodal reasoning as execution of a graph whose perception and logic nodes output conformal, set-valued predictions and whose controller allocates computation based on these certificates. This design grounds answers in verifiable evidence, reduces error compounding and hallucinations, and exposes a principled accuracy–compute trade-off. Empirically, PoP improves performance and reliability over strong chain-of-thought, ReAct-style, and program-of-thought baselines across document, chart, and multi-image QA while using computation more efficiently. PoP is model- and tool-agnostic and complements existing MLLMs by replacing single-valued decisions and heuristic control with stepwise guarantees and adaptive policies.

References

[1] Jean-Baptiste Alayrac et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022. 1

[2] Anastasios Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2021. 1

[3] Stephen Bates, Emmanuel J. Candès, Lihua Lei, and Yaniv Romano. Distribution-free, risk-controlling prediction sets. *Journal of the American Statistical Association*, 2023. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 6

[5] Xi Chen, Carlos Riquelme, Neil Houlsby, et al. Pali-x: On scaling up multilingual vision and language models, 2023. 1

[6] Yuning Du et al. Paddleocr: A practical ultra lightweight ocr system, 2020. 6

[7] Song Feng, Shreya Saxena, Khyathi Raghavi Chandu, et al. Multidoc2dial: Modeling dialogues grounded in multiple documents. In *EMNLP*, 2021. 6

[8] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 1

[9] Zhenmei Ji et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2023. 1

[10] Geewook Kim, Teakgyu Ryu, Heeyoung Ahn, et al. Donut: Document understanding transformer without ocr. In *ECCV*, 2022. 1

[11] Sehoon Kim et al. Chartpot: Program-of-thought for chart understanding. In *CVPR*, 2024. 1, 6

[12] Jiahui Lee, Chenliang Li, Hieu Pham Nguyen, et al. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *ICML*, 2023. 1

[13] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 2018. 2

[14] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 1

[15] Haotian Liu et al. Llava-1.5: Improved baselines and evaluation for language-vision models. In *NeurIPS Datasets and Benchmarks*, 2024. 1

[16] Pengfei Liu et al. Towards chain-of-thought reasoning in multimodal models, 2023. 1

[17] Ahmed Masry, Nikolaos Pappas, et al. Chartqa: A benchmark for question answering on charts. In *ACL*, 2022. 1, 6

[18] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for document visual question answering. In *WACV*, 2021. 6

[19] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Infographicvqa. In *CVPR*, 2022. 1, 6

[20] Yulei Nie et al. Program-of-thought prompting for visual question answering. In *ICCV*, 2023. 1

[21] Yash Patel, Vatsal Pahuja, Yash Patel, et al. Textcaps: A dataset for image captioning with reading comprehension. In *ECCV*, 2020. 6

[22] Baolin Peng et al. Kosmos-2: Grounding multimodal large language models to the world, 2023. 1

[23] Zejiang Shen, Ruochen Zhang, Micah Dell, Jacob Lee, Zhongliang Ren, Yichong Xu, Ze Sun, Junting Wang, Yue Jiang, Zhongliang Li, et al. Layoutparser: A unified toolkit for deep learning based document image analysis. In *ICDAR*, 2021. 6

[24] Amanpreet Singh, Vivek Natarajan, Meet Shah, Xinlei Jiang, Dhruv Chen, Yashas Wu, Dhruv Batra, Devi Parikh, Anna Rohrbach, and Kevin Shih. Textvqa: Towards question answering on text-rich images. In *CVPR*, 2019. 6

[25] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005. 1

[26] An Wang et al. Qwen2-vl: Enhancing vision-language models with better perception and reasoning, 2024. 1

[27] Junnan Yang et al. Mm-react: Prompting chatgpt for multimodal reasoning and action, 2023. 1

[28] Shunyu Yao et al. React: Synergizing reasoning and acting in language models. In *NeurIPS*, 2022. 1

[29] Denny Zhu et al. Multimodal chain-of-thought reasoning in language models. In *NeurIPS Workshop on Reasoning*, 2023. 1