frontiers in Computer Science

TYPE Original Research
PUBLISHED 17 June 2026
DOI 10.3389/fcomp.2026.1817034

OPEN ACCESS

EDITED BY
Tailia Malloy,
Universite du Luxembourg - Campus Kirchberg, Luxembourg

REVIEWED BY
Kenza Amara,
ETH Zürich, Switzerland
Xuesong Zhang,
Hefei University of Technology, China

*CORRESPONDENCE
Christopher Myers
christopher.myers.29@us.af.mil

RECEIVED 25 February 2026
REVISED 21 April 2026
ACCEPTED 19 May 2026
PUBLISHED 17 June 2026

CITATION
Sridhar S, Ganesh SS, Bajaj G, Myers C and Parthasarathy S (2026) Identifying knowledge gaps on the edge for visual question answering.
Front. Comput. Sci. 8:1817034.
doi: 10.3389/fcomp.2026.1817034

Identifying knowledge gaps on the edge for visual question answering

Sarikaa Sridhar[1], Sriram Sai Ganesh[1,2], Goonmeet Bajaj[1,3], Christopher Myers[1,4]* and Srinivasan Parthasarathy[1]

[1]Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, United States, [2]Department of Computer Science, Johns Hopkins University, Baltimore, MD, United States, [3]Amazon AGI, Boston, MA, United States, [4]Air Force Research Laboratory, Dayton, OH, United States

Human Cognition is complex and highly sophisticated system capable of accomplishing astonishing feats, partly due to our ability to seek out and overcome "unknowns," or gaps in our knowledge, skills, and capabilities. Artificial Intelligence (AI) systems often draw inspiration from human intelligence, however, they often lack the ability to recognize when their knowledge is insufficient, leading them to provide answers even when incorrect. This limitation poses significant challenges, particularly in human-AI teaming and edge AI scenarios, where systems may lack requisite knowledge of the environment. To address this, we propose Tiny Knowledge Gap Identification (TinyKGI), a lightweight framework for automatically identifying plausible cognitive skills that the model lacks (i.e., Knowledge Gaps; KGs), which could lead to incorrect predictions. Our framework leverages human cognitive skills to structure how AI models reason and to define the types of Knowledge Gaps they may plausibly exhibit. By identifying insufficient cognitive capabilities, TinyKGI enables the development of more reliable and robust AI systems. TinyKGI uses a deep learning approach to classify different types of KGs for multimodal reasoning tasks while enabling efficient inference in resource-constrained environments. Through model quantization, we significantly reduce the memory footprint and execution time, resulting in a compact and lightweight model. Evaluated on three datasets, TinyKGI improves Macro-F1 scores by up to 10% over the previous state-of-the-art, while achieving up to a 1.8× speedup and a 4× reduction in memory usage. We further evaluate TinyKGI on an edge device (Jetson Nano), where it achieves a 1.7× speedup and a 3.9× reduction in memory with only a 0.1% degradation in Macro-F1 score.

KEYWORDS
artificial intelligence, cognitive science, edge computing, human-AI teaming, multi-modal learning

1 Introduction

Autonomous AI systems are increasingly being used and deployed for a wide range of tasks and on edge devices. Research on human cognition and AI systems has been ongoing for many years, and the two are deeply interconnected. Humans can process and perform tasks through cognitive processes that encompass capabilities such as perception, memory, reasoning, etc. (Carroll, 1993). They also learn from experience and accumulate knowledge that can be used for new tasks (Spelke, 2017). These abilities allow humans to recognize when they do not know something and seek more information to complete

the task, adapt their strategies (Newell et al., 1972). Humans further leverage metaknowledge using patterns of past knowledge as sources of information to guide problem solving (Gentner and Collins, 1981; Gonzalez and Dutt, 2011).

Autonomous AI systems, on the contrary, lack the ability to automatically adapt to changing environments. Unlike humans, who can identify gaps in their knowledge and employ various approaches to resolve them (Newell et al., 1972), AI models frequently produce confident but incorrect answers when faced with incomplete, ambiguous, or insufficient information. Research in human cognition and AI has long progressed and complements each other. There have been studies of AI systems inspired by human cognition, such as research for AI systems to learn to build causal models to support explanation and understanding, ground learning, enrich knowledge and harness compositionality to rapidly learn new tasks (Lake et al., 2017). Indeed, the influence cognitive science research has on AI is difficult to overstate. While deep learning frameworks have become increasingly powerful, they still lack the ability to identify when essential knowledge is missing.

Knowledge is gradually accumulated through experience and learning across humans' lifespan, allowing them to apply prior understanding to new tasks (Spelke, 2017) and it is unreasonable to expect humans or intelligent systems to contain complete knowledge about every possible situation. Additional knowledge may be acquired through explicit instructions or interaction to guide to learn to perform new tasks, but this knowledge may also be insufficient or incomplete. Such insufficiency can cause an intelligent system to behave erroneously and giving rise to one or more Knowledge Gaps. Despite the importance of such gaps for understanding and improving intelligent behavior, there is relatively little research that directly focuses on identifying and characterizing them.

Knowledge gaps (Schmidt, 2020) occur when there is limited or missing information, leading the system to cause erroneous behavior, process halting, poor contextual reasoning, and grounding. Motivation for addressing KGs comes from how humans process and reason through tasks to help models learn to address these gaps (Bajaj et al., 2022; Motlagh et al., 2022; Ball et al., 2010). Our goal is to incorporate human cognitive abilities into autonomous AI systems to make them more robust and reliable by mapping cognitive skills to KGs required for the task. Identifying the knowledge gap type present in the system can provide actionable insights for improving the system's reasoning capabilities. We use human cognitive skills as a foundation to define and interpret Knowledge Gaps in AI systems. At the first level of interaction, our framework draws directly from human cognitive skills to structure how AI models reason and to define the types of Knowledge Gaps they may plausibly exhibit.

To study this problem, researchers have investigated Visual Question Answering (VQA) as a testbed. In recent years, VQA (Antol et al., 2015) has emerged as a powerful and widespread application of the ability of AI systems to assimilate knowledge sources and use them to draw inferences. This AI-complete (Özdemir and Akagündüz, 2024) task is not feasible to accomplish with any single algorithm, instead it is situated at the intersection of three areas of cognitive science & AI: *reasoning*, *language*, and *vision* (Bajaj, 2024). Recent multimodal models like GPT-4 (OpenAI et al., 2024) and Gemini (Team et al., 2024) show remarkable capabilities in processing visual and language data for VQA (Liu M. et al., 2024; Özdemir and Akagündüz, 2024); however, robustness is still compromised when faced with uncertain, ill-defined, or ambiguous inputs. Prior work has examined this issue through uncertainty estimation in Visual Question Answering, including approaches such as Zhang et al. (2024), which are closely aligned with the motivation of addressing overconfident but incorrect predictions. These studies identify overconfidence in VQA models, often arising from language bias, and propose calibration techniques to improve reliability and generalization. While such approaches focus on identifying when a model may be incorrect, our work focuses on identifying the underlying knowledge gaps required for a given task, providing insight into why such errors occur.

Prior research (Bajaj et al., 2022) in Knowledge Gap Identification (KGI) for VQA relies on rule-based methods and metadata associated with questions to classify knowledge gap types. Although Bajaj et al. (2022) has proposed the KG processes and made a huge effort toward the framework implementation, it still limits in performance and scalability. Table 1 presents the Knowledge Gaps and their corresponding cognitive capabilities, adapted from the taxonomy in Schmidt (2020) and Bajaj et al. (2022), which was derived from prior cognitive science research (Collins et al., 1975; Gentner and Collins, 1981).

AI systems are increasingly being deployed on edge devices and shifting rapidly from cloud-based infrastructures to on-device computation. This shift is motivated by the demand for lower latency in human-AI interaction, energy efficiency, scalability, robustness, and enhanced privacy. However, implementing deep neural networks for feature extraction and KGI model execution in resource-constrained environments like edge devices remains an unsolved challenge due to computational overhead.With the exponential growth of IoT devices producing data daily, there is an increasing need for on-device AI that is faster, low-power and always available. Performing inference on-device near the data source enables faster response times and preserves privacy while reducing the energy cost and overhead associated with wireless communication. Edge inference is particularly beneficial for applications where real-time decision-making is crucial, such as autonomous systems.

In this work, we utilize a combination of pre-trained language models (e.g., BERT, MPNet, and ALBERT) for text and foundation vision models (e.g., SAM, SAM2, CLIP, and DINOv2) for images. These foundation models form the backbone of the TinyKGI framework, enabling robust linguistic and visual feature extraction, which allows efficient identification of knowledge gaps. To overcome the limitations of the prior research, this work presents Tiny Knowledge Gap Identification, an efficient and compact model leveraging pre-trained language and foundation vision models to identify knowledge gaps in VQA reasoning. At the second level of interaction, TinyKGI supports interaction between humans and on-device AI by enabling KGI inference for a given task in resource-constrained settings. TinyKGI consists of three components a question encoder, an image encoder and classifier model which integrates multi-modal features for KG identification. Adapting our

TABLE 1 Knowledge gap types and associated cognitive capabilities.

Knowledge gap type	Cognitive capabilities	Knowledge gap description
Activity	Perception	Actions being performed by a person or object cannot be interpreted.
Material	Perception	The object's composition or texture is not clear.
State	Perception, categorization	The object's or person's condition cannot be interpreted.
Location	Spatial, perception, attention	The physical location of a place cannot be understood.
Reasoning	Reasoning, memory	Unable to perform logical inferences or draw conclusions due to insufficient knowledge.
Attribute	Perception	The object's properties are unknown.
Sentiment	Language, Emotion	The sentiment conveyed cannot be interpreted.
Counting	Reasoning, perception, attention	The quantity of the item cannot be estimated.
Entity resolution	Attention, perception	The object's presence cannot be interpreted.
Direction/positional reasoning	Spatial, Attention, Perception	The object's position is unsure.
Size	Spatial, attention, perception	The object's size or distance cannot be interpreted.
Color	Perception	The color of the object is unclear.
Scene recognition	Attention, perception	The interpretation of the scene is unclear.
Utility affordance	Memory, reasoning	The object's utilities and affordances cannot be determined.

KGI framework for resource-constrained environments enables real-time identification of knowledge gaps directly on edge devices, making the system faster, energy-efficient, privacy-preserving, and reliable. While the current work demonstrates inference under simulated resource-constrained settings, our long-term goal is to achieve on-device deployment for robust and scalable applications. The main contributions are as follows:

- TinyKGI model: We identified the efficient approach for automatic KG identification.
- Incorporating image features: While the prior work uses only question features, we explored adding image features for KG identification.
- Inference on resource-constrained environment: Tiny Machine Learning techniques are applied to the encoders and model to perform inference on edge devices.
- Inference on an edge device: We evaluate TinyKGI on a Jetson Nano, demonstrating practical efficiency gains with minimal impact on accuracy.

2 Related work

Autonomous AI systems are said to have a plausible set of Knowledge Gaps (KGs) (Schmidt, 2020) tailored to the task being solved. Schmidt introduced the KG taxonomy that composes the types of KGs in a hierarchical structure. This taxonomy consists of various KGs retrieved from prior research in cognitive science and tailored for these systems. Figure 1 shows the KG taxonomy used in this work and is not a fixed or exhaustive list of KGs.

Further, Bajaj et al. (2020) extended the KG taxonomy and proposed a refined version specifically for Visual Question Answering, identifying KGs for reasoning over language and visual inputs. This updated version of the KG taxonomy for VQA task is adapted in our work. Additionally, the author tags one or more KGs to the questions to quantify the VQA model performance with respect to each of the knowledge gaps. Bajaj et al. (2022) proposed a paradigm for KG detection, identification and resolution processes. This work uses metadata present in VQA datasets such as question type or functional programs to annotate KGs. For example, if the question type was "positionVerify," "Direction Gap" is assigned as it represents identifying spatial position. A Bidirectional Encoder Representation from Transformers (BERT) classifier model was fine-tuned for this annotated dataset to predict knowledge gaps.

The Tiny Machine Learning paradigm aims to implement methods to improve model performance by less computation, less data for training, and inference. Mainly developed to build smaller models and deploy those models on edge devices to perform inference on low power hardware. The article by Jacob et al. (2018) proposes a quantization scheme to perform inference using integer-only arithmetic by developing a approach where the model is trained in floating point and integer inference. Lin et al. (2022) discusses training AI models on edge devices by using two methods Quantization-aware Scaling and Sparse Update for reducing the computation during model training and performing inference after code generation. The article by Liu Z. et al. (2024) proposes a solution to reduce large quantization errors when post-training quantization (PTQ) is applied to weights, activations by learning orthogonal rotation matrices.

3 Visual question answering datasets

Numerous VQA datasets are available in the literature, including CLEVR (Johnson et al., 2017), GQA (Hudson and Manning, 2019), VQA (Antol et al., 2015), VQA v2.0 (Goyal et al., 2017), TDIUC (Kafle and Kanan, 2017), and OK-VQA (Marino et al., 2019). VQA datasets typically contain images, questions, and answers for model training. For Knowledge Gap Identification, we require the KGs for each question-image

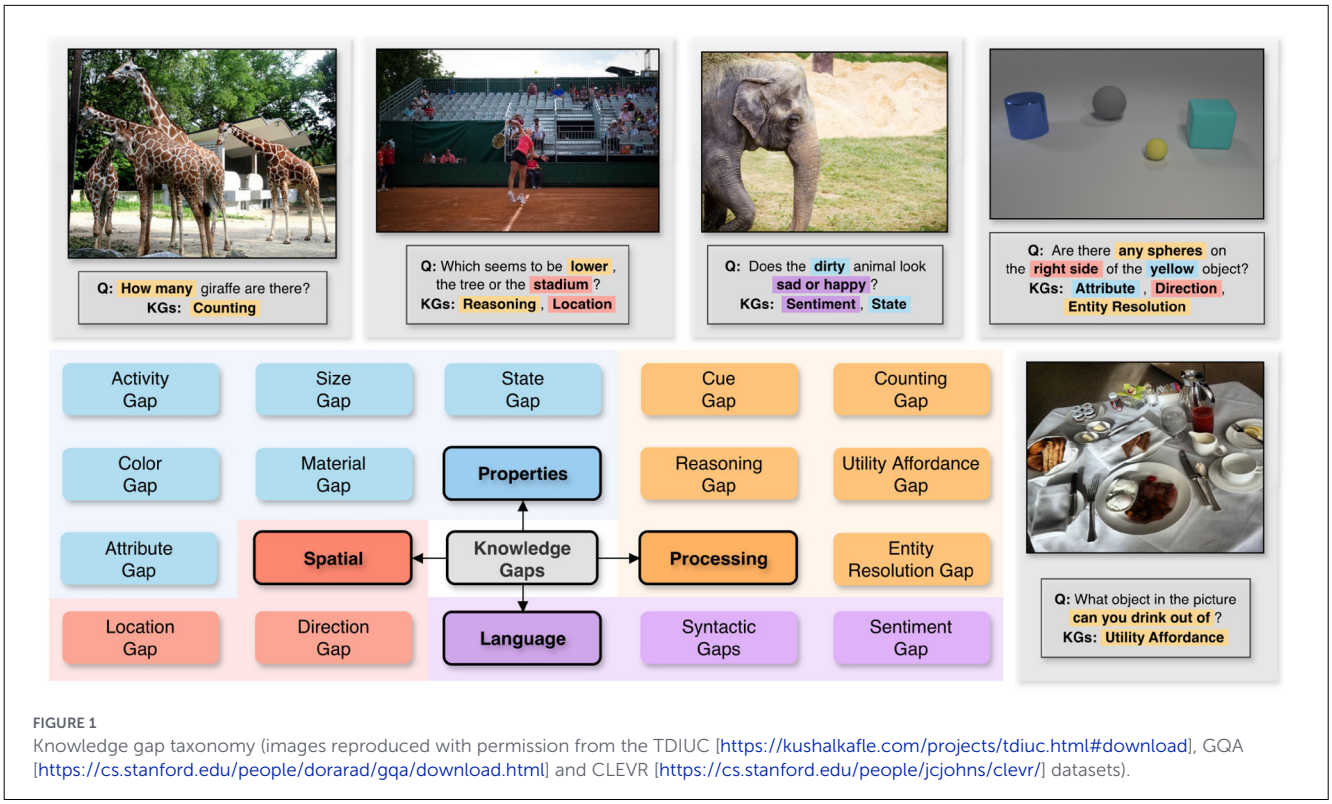

FIGURE 1
Knowledge gap taxonomy (images reproduced with permission from the TDIUC [https://kushalkafle.com/projects/tdiuc.html#download], GQA [https://cs.stanford.edu/people/dorarad/gqa/download.html] and CLEVR [https://cs.stanford.edu/people/jcjohns/clevr/] datasets).

pair. The VQA datasets used in this work are: GQA, CLEVR, and TDIUC. For the datasets that do not come with KGs as metadata, rule-based mapping from Bajaj et al. (2020) is used to tag the KG to question-image pair. Table 2 shows the dataset and its corresponding plausible KGs shortlisted for the nature of questions and images. The Compositional Language and Elementary Visual Reasoning (CLEVR) contains synthetic images composed of 3D objects covering a range of reasoning abilities. To evaluate our TinyKGI pipeline, we use CLEVR: a diagnostic dataset with detailed annotations and minimal biases (Johnson et al., 2017). Further, we test the generalization capability of our approach on real-world images using GQA (Hudson and Manning, 2019) and its rich scene graph annotations. Both of these datasets have one or more KGs for each question-image pair. Next, is the Task Directed Image Understanding Challenge (TDIUC) dataset (Kafle and Kanan, 2017), which consists of question-image pairs and their type or the reasoning skill needed for prediction. It has one KG per question-image pair.

TABLE 2 Knowledge gap tags for VQA datasets.

VQA datasets	Knowledge gap tags
GQA	Activity, attribute, direction, entity resolution, location, material, reasoning, sentiment, size, state
CLEVR	Attribute, counting, direction, entity resolution, material, size
TDIUC	Object presence, color, scene recognition, counting, attribute, activity recognition, positional reasoning, object recognition, utility affordance, sentiment understanding

4 Methodology

4.1 Tiny knowledge gap identification

Let the input to the TinyKGI model be denoted as (Q, I) where Q represents the natural language question and I represent the image. The knowledge gaps for the VQA datasets are denoted as $K = \{k_1, k_2, k_3.., k_n\}$. The aim of knowledge gap identification is to predict the plausible subset of K knowledge gaps leading to incorrect prediction. The proposed TinyKGI framework consists of three components:

1. Question encoder: Uses pretrained language transformer models to extract the feature embeddings from Q and produces a fixed-length embedding via a pooling operation over the last-layer token representations, resulting in the question feature vector $f_{\mathcal{Q}}$.
2. Image encoder: Uses pretrained vision transformers and convolution neural networks to extract image feature vector denoted as $f_{\mathcal{I}}$.
3. Classifier and fusion mechanism: We train two TinyKGI models: question-only (TinyKGI-Q) and question-image model (TinyKGI-Q+I). TinyKGI-Q uses the question features $f_{\mathcal{Q}}$ as input. TinyKGI-Q+I model concatenates $f_{\mathcal{Q}}$ and $f_{\mathcal{I}}$ into $F_{\mathcal{Q}+\mathcal{I}} = [f_{\mathcal{Q}} \parallel f_{\mathcal{I}}]$. The TinyKGI model is implemented using a Multi-Layer Perceptron (MLP) for classification.

Figure 2 shows the overall TinyKGI model architecture and it is systematically benchmarked using different question and image feature encoders to determine the optimal configuration. For GQA and CLEVR datasets we implement multi-label classification to identify KGs and for TDIUC multi-class classification is implemented.

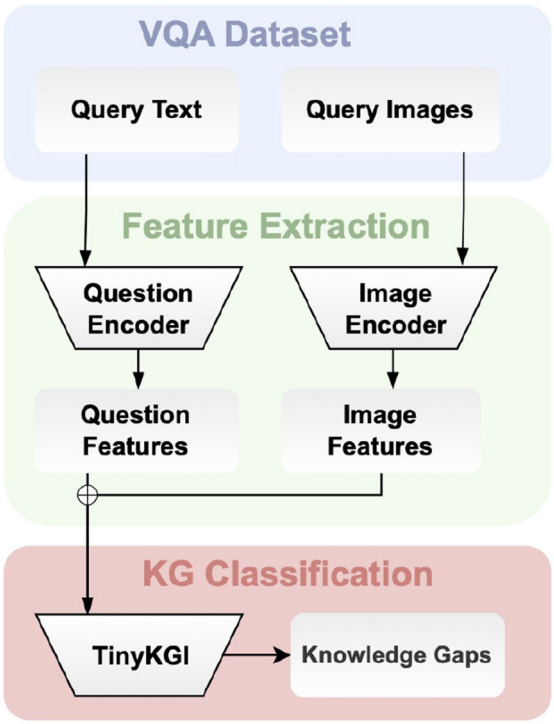

FIGURE 2
TinyKGI architecture.

4.2 Question feature extraction

This section formalizes the derivation of the question representation $f_Q \in \mathbb{R}^{768}$ that is supplied to the TinyKGI model.

4.2.1 Sentence-transformer encoder

Each question is tokenized into a WordPiece/SentencePiece sequence $Q = (w_1, \ldots, w_L)$, $L \leq L_{\max}$. A pre-trained language model from the `sentence_transformers` library is used for the final mapping

$$f_\theta : \mathcal{V}^{\leq L_{\max}} \longrightarrow f_Q \in \mathbb{R}^{768}$$

of the sequence to a fixed-length question feature vector.

All text encoders are invoked via the `sentence_transformers` library (v4.1) (Reimers and Gurevych, 2019).

TABLE 3 Sentence-Transformer checkpoints evaluated in the ablation study.

`paraphrase-albert-small-v2`
`paraphrase-MiniLM-L3-v2`
`all-MiniLM-L6-v2`
`all-MiniLM-L12-v2`
`multi-qa-distilbert-cos-v1`
`all-distilroberta-v1`
`all-mpnet-base-v2`
`distiluse-base-multilingual-cased-v1`
`paraphrase-multilingual-mpnet-base-v2`

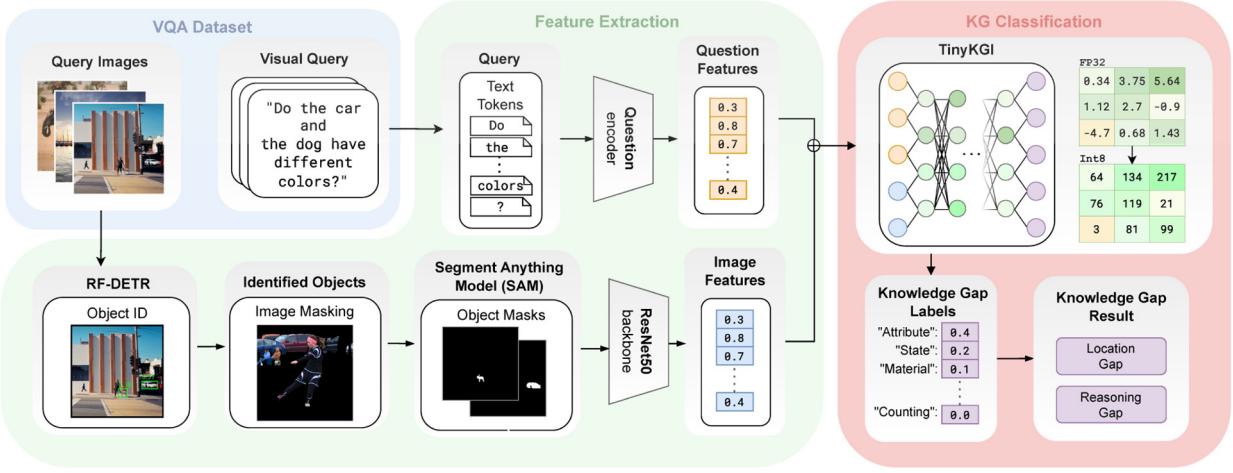

FIGURE 3
Knowledge gap identification and TinyKGI model architecture using segment anything (SAM) model.

4.2.2 Model zoo and selection rule

We instantiate f_θ with the nine publicly available checkpoints shown in Table 3. We retain the four encoders that either (a) achieve the highest macro-F1 or (b) run inference with the smallest latency or memory footprint:

- BERT-base-uncased,
- all-mpnet-base-v2 (best accuracy/speed trade-off),
- paraphrase-MiniLM-L3-v2 (fastest), and
- paraphrase-albert-small-v2 (smallest memory).

These four frozen-weight models serve as drop-in modules for the full Experimental Results section and additional results in Supplementary section.

4.3 Image feature extraction

Let an RGB image be denoted by $I \in \mathbb{R}^{H\times W\times 3}$, where H and W are the original height and width, respectively. For every question-image pair, we consider three approaches to computing visual representations:

4.3.1 Global features

We experiment with two ImageNet-1K pretrained backbones: ResNet-18 and ResNet-34, following the original architecture definitions (He et al., 2015). We remove the classification head for TinyKGI image feature extraction. The final `avgpool` layer outputs a $C = 2048$-dimensional embedding for both ResNet-18 and ResNet-34 models.

The input image I is bilinearly resized & cropped to $\tilde{I} \in \mathbb{R}^{224\times 224\times 3}$. Pixel intensities are rescaled to $[0, 1]$ and channel-wise normalized using $\boldsymbol{\mu} = [0.485, 0.456, 0.406]$ and $\boldsymbol{\sigma} = [0.229, 0.224, 0.225]$.

The pre-processed crop $\tilde{I}$ is passed through the chosen backbone in inference mode, producing its penultimate-layer activation. The resulting image feature vector $f_\mathcal{I} \in \mathbb{R}^{2048}$ vector encapsulates holistic scene content and serves as the baseline visual descriptor.

4.3.2 Grounded localized features

In order to focus on question-relevant visual evidence, we derive an image representation grounded in localized features. We isolate regions of an image that are related to the visual question using an open-set object detector, Grounded SAM (Ren et al., 2024), based on Meta's Segment Anything Model (SAM) (Kirillov et al., 2023).

Given a visual query Q as a caption and its corresponding image I as input, Grounded SAM returns a set of K binary masks

$$\mathcal{M} = \{M_k\}_{k=1}^{K}, \text{ where each mask } M_k \in \{0, 1\}^{H\times W}$$

For each mask M_k we compute its axis-aligned bounding box

$$B_k = (x_{\min}^{(k)}, y_{\min}^{(k)}, x_{\max}^{(k)}, y_{\max}^{(k)}).$$

The grounded region of interest is the tightest box enclosing all objects:

$$B^\star = \left(\min_k x_{\min}^{(k)},\ \min_k y_{\min}^{(k)},\ \max_k x_{\max}^{(k)},\ \max_k y_{\max}^{(k)}\right).$$

We crop I to $B^\star$ and resize the crop to 224 × 224px, obtaining $I^\star$.

Applying the frozen ResNet backbone again yields the image feature vector $f_\mathcal{I} \in \mathbb{R}^{2048}$ that aims to capture localized visual features from image regions pertinent to each query.

The joint image-question representation consumed by the classifier is the row-wise concatenation of the question and image feature vectors

$$F_{\mathcal{Q}+\mathcal{I}} = [f_\mathcal{Q} \,\|\, f_\mathcal{I}] \in \mathbb{R}^{(768+d_I)}.$$

Figure 3 illustrates the TinyKGI architecture, where image features are extracted using the Segment Anything Model (SAM/SAM2) and subsequently used for the downstream Knowledge Gap Identification task.

4.3.3 Vision transformer-based global features

In addition to convolutional backbones, we also evaluate vision transformer–based image encoders to extract global visual representations. Specifically, we experiment with CLIP (Radford et al., 2021), DINOv2 (Oquab et al., 2023), and Google's Vision Transformer (ViT) (Dosovitskiy, 2020) models pre-trained on large-scale image corpora.

For these models, the input image I is resized to 224 × 224 pixels and normalized according to the preprocessing requirements of each pre-trained backbone. Image features are extracted from the final transformer layer by using the pooled output or the `[CLS]` token representation, depending on the architecture. This produces a fixed-length image embedding $f_\mathcal{I} \in \mathbb{R}^{d_I}$ that captures global semantic information from the image.

These vision transformer–based representations are used directly for Knowledge Gap Identification by concatenation with the corresponding question embedding, forming the joint representation $F_{\mathcal{Q}+\mathcal{I}} = [f_\mathcal{Q} \,\|\, f_\mathcal{I}]$. Unlike grounded localized features, these models operate on the full image and do not require visual query or segmentation masks.

4.4 TinyKGI classifier

The TinyKGI classifier is implemented as a lightweight multi-layer perceptron (MLP) designed to predict Knowledge Gap (KG) labels from question and, optionally, image representations. The classifier operates on either question-only features (TinyKGI-Q) or concatenated question–image features (TinyKGI-Q+I), depending on the configuration.

Let $f_\mathcal{Q} \in \mathbb{R}^{d_Q}$ denote the question feature vector produced by the question encoder and $f_\mathcal{I} \in \mathbb{R}^{d_I}$ denote the image feature vector

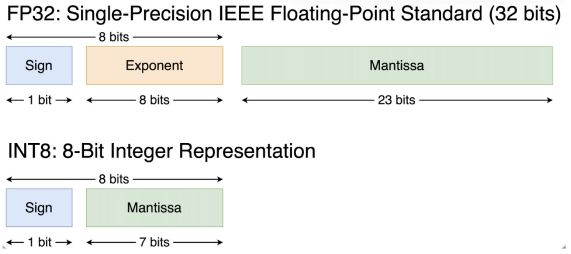

FIGURE 4
Floating point to integer quantization.

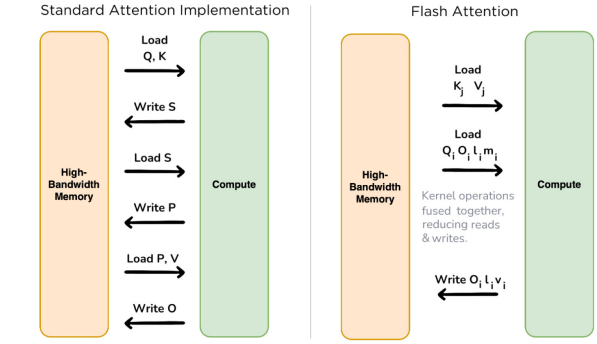

FIGURE 5
Flash attention.

produced by the image encoder. For the multimodal setting, the input to the classifier is formed by concatenation:

$$x = [f_Q \parallel f_I] \in \mathbb{R}^{d_Q + d_I}.$$

In the question-only setting, the classifier input reduces to $x = f_Q$.

The classifier consists of two fully connected hidden layers with ReLU activations, followed by an output layer that predicts the Knowledge Gap labels. The network architecture is defined as:

$$h_1 = \text{ReLU}(W_1 x + b_1),$$
$$h_2 = \text{ReLU}(W_2 h_1 + b_2),$$
$$\hat{y} = W_3 h_2 + b_3,$$

where the first and second hidden layers have dimensions 512 and 256, respectively. Dropout with a rate of 0.3 is applied after each hidden layer to mitigate overfitting.

For datasets requiring multi-label Knowledge Gap prediction (GQA and CLEVR), the output logits $\hat{y}$ are passed through a sigmoid activation. For datasets with mutually exclusive Knowledge Gap labels (TDIUC), a softmax activation is applied. The TinyKGI classifier is trained using the binary cross-entropy loss with logits (`BCEWithLogitsLoss`) for all datasets. Model parameters are optimized using the AdamW optimizer, which decouples weight decay from gradient updates to improve generalization.

4.5 Tiny machine learning optimizations

To enable efficient deployment of TinyKGI in resource-constrained and edge environments, we apply Tiny Machine Learning (TinyML) optimization techniques to both the feature encoders and the downstream Knowledge Gap Identification (KGI) classifier. These optimizations reduce inference latency and memory footprint while preserving performance.

4.5.1 Post-training quantization

We apply post-training dynamic quantization from pytorch library to the pre-trained question encoders, image encoders, and the TinyKGI classifier. In this setting, model weights are quantized from 32-bit floating-point (`fp32`) to 8-bit integer (`int8`) precision, while activations are computed dynamically at inference time as shown in Figure 4. This approach significantly reduces memory consumption and improves inference efficiency without requiring retraining or access to calibration data.

For transformer-based encoders, quantization is applied to the linear projection layers within both the feed-forward and attention blocks. For the TinyKGI classifier, all linear layers are quantized to `int8`, enabling lightweight downstream inference on CPU.

4.5.2 Post-training quantization with flash attention

For transformer-based encoders, we further combine post-training quantization with Flash Attention (Dao, 2023). Flash Attention replaces the standard scaled dot-product attention mechanism with a fused, memory-efficient kernel that reduces memory access overhead and improves computational throughput. Since the attention mechanism accounts for a significant portion of inference latency in Vision Transformer (ViT)-based and language transformer models, enabling Flash Attention provides substantial speedups during inference.

Meta's Segment Anything Model (SAM) (Kirillov et al., 2023) employs a Vision Transformer-based (ViT-based) image encoder to assimilate image features. The inference cost of the attention mechanism is over 90% of the total runtime to compute masks for objects in an image. We enable Flash Attention 2 (Dao, 2023) (Figure 5) replacing naive Scaled Dot Product Attention to achieve significantly higher throughput at inference time per image.

4.5.3 SpinQuant

We additionally evaluate SpinQuant (Liu Z. et al., 2024), a rotation-based post-training quantization technique designed to improve quantization robustness by mitigating weights and activation outliers. SpinQuant introduces learnable orthogonal rotation matrices into transformer layers and attention blocks prior to quantization. These rotations are trained to redistribute activation values, reducing quantization loss in low-precision inference.

In our implementation, the rotation matrices are trained using a combination of mean squared error (MSE) and contrastive loss, with early stopping applied to prevent overfitting. While SpinQuant effectively reduces model size and maintains stable performance, its benefits are more pronounced in tasks with significant activation outliers. As shown in our experiments, Knowledge Gap Identification does not strongly exhibit such characteristics, resulting in comparatively smaller performance gains relative to PTQ with Flash Attention.

5 Experimental results

5.1 Experimental setup

In this section, we evaluate the performance of our proposed TinyKGI framework on three Visual Question Answering datasets, with primary analysis conducted on the GQA dataset. Ablation study can be found in the Supplementary section. We compare question-only TinyKGI models (TinyKGI-Q) and question-image models (TinyKGI-Q+I) across multiple Knowledge Gaps using F1 score, inference time, and memory footprint as evaluation metrics.

The TinyKGI classifier is implemented as a Multi-Layer Perceptron (MLP) consisting of three fully connected layers with a dropout rate of 0.3. For multi-label classification tasks (GQA and CLEVR), a sigmoid activation function is used in the output layer, whereas a softmax activation is applied for multi-class classification in the case of TDIUC. All models are trained using Binary Cross-Entropy with logits loss.

Training is performed on H100 or V100 GPUs with a batch size of 64. Early stopping is applied with a patience of 10 epochs to prevent overfitting. Inference is performed on GPU to compare multiple optimization techniques.

We evaluate four pre-trained question encoders and seven image encoders, resulting in a comprehensive comparison across three datasets. To assess inference on resource-constrainted environments, we further apply multiple TinyML optimization techniques, including post-training quantization (PTQ), PTQ with Flash Attention, and SpinQuant.

5.2 Best knowledge gap identification model

Table 4 reports the Macro-F1 scores of the best-performing full-precision and fully quantized TinyKGI models across the three datasets. The full-precision model is trained and evaluated using 32-bit floating-point arithmetic, whereas the fully quantized model performs both training and inference using 8-bit integer precision. Overall, quantized TinyKGI achieves consistently high performance, with only a marginal decrease in Macro-F1 observed when moving from full-precision to int8 quantized inference.

These results indicate that TinyKGI is robust to quantization, enabling efficient deployment on resource-constrained devices while preserving accuracy. Notably, for CLEVR, both the full-precision and quantized models achieve perfect Macro-F1 performance, highlighting the effectiveness of the proposed approach across different datasets.

TABLE 4 Macro-F1 scores for the best full-precision and quantized question-only models across datasets.

Dataset	TinyKGI (Mpnet-base) (fp32)	TinyKGI (MiniLM-L3) (int8 + flash attention)
GQA	99.13	99.00
CLEVR	100.00	100.00
TDIUC	99.56	99.04

TABLE 5 Macro-F1 performance of TinyKGI question-only and question-image models across knowledge gaps on GQA.

Knowledge gap	Previous SOTA(fp32)	TinyKGI-Q (int8)	TinyKGI-Q+I (int8)
Activity	91.0	99.4	98.5
Attribute	96.0	99.5	99.1
Direction	96.0	98.3	98.1
Entity resolution	97.0	99.4	99.0
Location	79.0	99.2	99.0
Material	84.0	97.9	96.9
Reasoning	97.0	99.5	98.9
Sentiment	79.0	99.8	99.1
Size	86.0	99.3	98.3
State	83.0	97.7	95.8
Macro F1	**88.8**	**99.0**	**98.27**

5.3 Comparison with previous state-of-the-art

Table 5 compares the proposed TinyKGI models with the previous state-of-the-art Knowledge Gap Identification approach introduced by Bajaj et al. (2022), which uses BERT-based question feature fine-tuning. Figure 6 shows the comparison of the quantized TinyKGI-Q and TinyKGI-Q+I models with the previous state-of-the-art (FP32), highlighting performance improvements across individual knowledge gaps.

Our results demonstrate substantial improvements over the prior approach across all evaluated Knowledge Gaps. Both TinyKGI-Q and TinyKGI-Q+I models outperform the previous state-of-the-art, with the question-only TinyKGI-Q model achieving the highest overall Macro-F1 score. This increase in performance highlights the efficiency of the proposed architecture KG Identification. TinyKGI-Q performs best for Sentiment, Attribute, Reasoning, and Activity Knowledge Gaps, as these gaps can be attributed to well-defined words and phrases in the question.

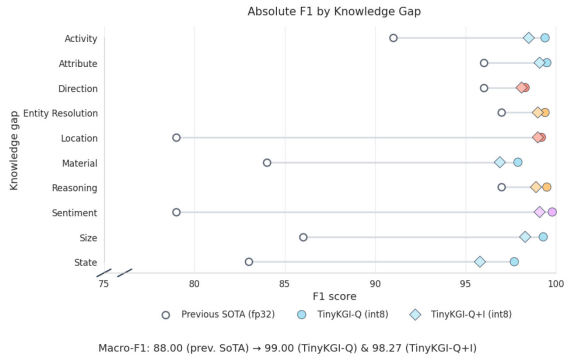

FIGURE 6
Absolute F1-score by knowledge gap.

5.4 Quantized TinyKGI-Q vs. TinyKGI-Q+I

Next, we analyze the performance of the question-image TinyKGI model (TinyKGI-Q+I), in which image and question embeddings are concatenated for Knowledge Gap Identification. Image features are extracted using a diverse set of encoders, including Segment Anything (SAM), SAM2, CLIP, Google ViT, DinoV2, and ResNet-34/18, covering both vision transformer-based and convolutional architectures.

Table 5 presents the Macro-F1 performance of the quantized TinyKGI-Q and TinyKGI-Q+I models. For all the Knowledge Gaps evaluated, incorporating image features does not improve performance for the Knowledge Gap Identification task. Instead, we consistently observe a decrease in Macro-F1 when comparing TinyKGI-Q+I to the corresponding question-only TinyKGI-Q models.

We attribute this behavior to the nature of the Knowledge Gap Identification task, which primarily depends on linguistic and semantic cues present in the question, rather than visual content. Unlike Visual Question Answering, where image understanding is essential for predicting answers, identifying the type of Knowledge Gap does not inherently require visual reasoning. Consequently, incorporating image features may introduce additional noise, negatively affecting classification performance.

Among the quantized models, the best-performing TinyKGI-Q configuration uses MiniLM-L3 as the question encoder, while the best TinyKGI-Q+I configuration combines MiniLM-L3 question features with ResNet-18 image features.

5.5 Efficiency and TinyML optimization

Table 6 reports the Macro-F1 performance of the TinyKGI-Q and TinyKGI-Q+I models under three optimization settings: full precision (FP32), post-training quantization with Flash Attention (PTQ + Flash Attention), and SpinQuant. Since Flash Attention improves inference efficiency without affecting model accuracy, we report results for PTQ with Flash Attention, as standard PTQ and PTQ with Flash Attention yield identical performance.

TABLE 6 Macro-F1 performance across different encoders and optimization techniques.

Model	FP32	PTQ + flash attention	SpinQuant
BERT	98.95	98.66	96.66
Mpnet-Base	99.13	98.97	98.03
MiniLM-L3	99.06	99.00	98.27
Albert-Small	99.05	98.68	97.28
SAM	97.91	96.45	95.43
SAM2	98.20	97.89	95.77
CLIP	97.85	97.65	97.31
DinoV2	97.04	96.83	95.47
ViT	98.02	98.02	97.21
ResNet34	98.26	98.24	–
ResNet18	98.27	98.19	–

Across all configurations, SpinQuant consistently achieves lower Macro-F1 scores compared to PTQ + Flash Attention. This behavior can be attributed to the nature of the Knowledge Gap Identification task, where we observe only a minimal fraction of extreme outliers or discretization characteristics that SpinQuant is designed to mitigate. As a result, the benefits of rotation-based quantization are limited in this setting. For the SpinQuant implementation, learnable orthogonal rotation matrices are inserted into the transformer layers and attention blocks of both question and image encoders. Training each SpinQuant model requires approximately five hours, with early stopping applied using a patience of five epochs.

Figure 7 shows a scatter plot illustrating inference speedup relative to the full-precision (FP32) baseline versus the corresponding drop in Macro-F1 score (%) across all question and image encoders under different TinyML optimization techniques, including PTQ (int8), PTQ with Flash Attention, and SpinQuant. In this plot, optimization strategies are represented using distinct marker shapes. We observe that MiniLM-L3 with PTQ + Flash Attention achieves the highest inference speedup of approximately 3 × with only a negligible drop in Macro-F1. This is followed by MiniLM-L3 with PTQ (int8), which achieves approximately 2.7 × speedup relative to FP32. Across most encoder architectures, PTQ + Flash Attention consistently yields higher speedups than standard PTQ (int8). Furthermore, both PTQ (int8) and PTQ + Flash Attention maintain a Macro-F1 drop below 0.5% while achieving speedups ranging from approximately 1.2 × to 3 × compared to the full-precision baseline, indicating a optimal efficiency–accuracy trade-off.

Across vision transformer–based encoders such as CLIP, DinoV2, and ViT, post-training quantization (PTQ) consistently reduces inference time compared to full-precision execution, with further improvements observed when combined with Flash Attention. In particular, PTQ with Flash Attention provides the largest speedups for transformer-based image encoders. SpinQuant yields smaller inference-time improvements relative to PTQ with Flash Attention, likely due to the additional overhead introduced by rotation operations. In contrast, convolution-based models such as ResNet-34 and ResNet-18 exhibit limited benefits from

Flash Attention and SpinQuant, as these techniques are primarily optimized for transformer architectures.

Figure 8 shows a scatter plot illustrating memory reduction relative to the full-precision (FP32) model versus the corresponding drop in Macro-F1 score (%) across all encoders, with different optimization techniques represented using distinct marker shapes. For our best-performing model, MiniLM-L3 with PTQ (int8), we observe approximately a 1.3× reduction in memory with only a marginal drop in Macro-F1. Additionally, for image-based models, PTQ (int8) achieves higher memory reductions of up to approximately 3.8×, while maintaining less than 0.4% drop in Macro-F1 score. We also observe that image models, including both CNN-based architectures such as ResNet and transformer-based models such as ViT, exhibit higher memory reduction. This can be attributed to the fact that image models contain a larger proportion of convolutional and linear layers, allowing quantization to be applied to a greater fraction of model parameters, resulting in higher memory reduction.

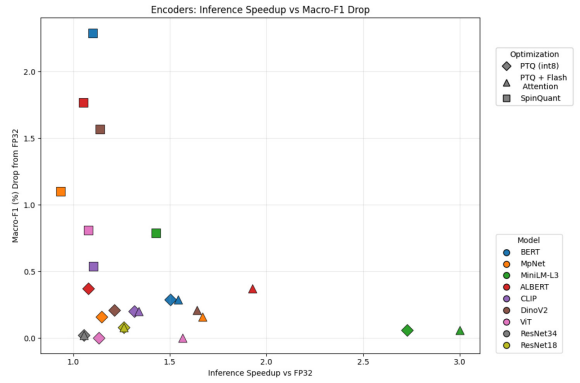

FIGURE 7
Inference speedup vs. Macro-F1 drop plot for question and image encoders.

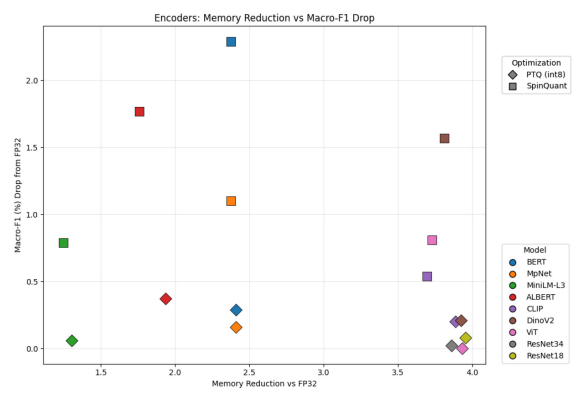

FIGURE 8
Memory reduction vs. Macro-F1 drop plot for question and image encoders.

We extract grounded localized image features using the SAM and SAM2 pipelines. The complete pipeline, from object detection to feature extraction, takes approximately 0.53 seconds per image for SAM and 0.3 seconds per image for the SAM2 PTQ with flash attention models, resulting in about 25%–45% speedup in throughput. Due to the multi-step nature of SAM/SAM2-based feature extraction, we report only the end-to-end inference time. Despite using grounded localized image features, we do not observe an improvement in TinyKGI performance compared to ResNet-based image encoders.

Figure 9 presents plots for TinyKGI-Q and TinyKGI-Q+I models as well. From these plots, we summarize the end-to-end inference time and memory footprint of the best-performing configurations. For the TinyKGI-Q model, we observe an overall inference speedup of approximately 3× along with a 1.3× reduction in memory usage compared to the full-precision baseline. Similarly, the best TinyKGI-Q+I model achieves an end-to-end inference speedup of approximately 1.5× and a memory reduction of about 1.8×. With these efficiency gains, PTQ (int8) results in only a 0.1% drop in Macro-F1.These results demonstrate that TinyKGI enables efficient, end-to-end Knowledge Gap Identification while maintaining strong performance under resource constraints. Figures 10–14 provides detailed per-encoder results, while the scatter plots summarize the overall efficiency–accuracy trade-offs.

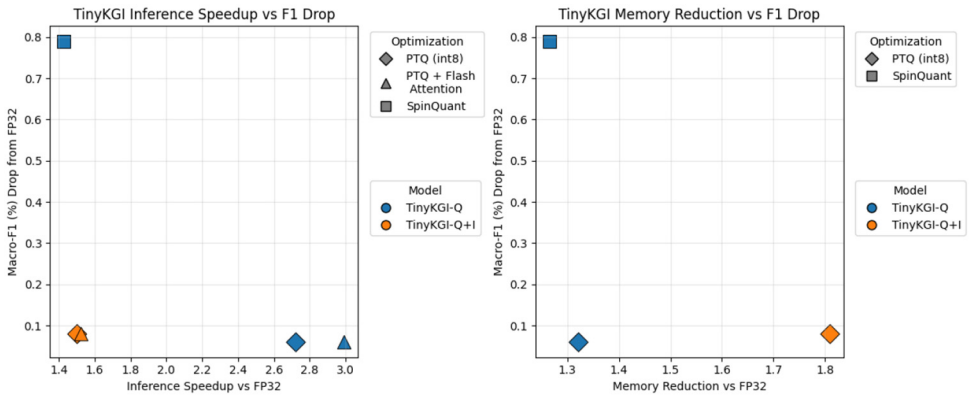

FIGURE 9
TinyKGI end-to-end inference time speedup and memory reduction across optimization techniques vs. macro-f1 drop.

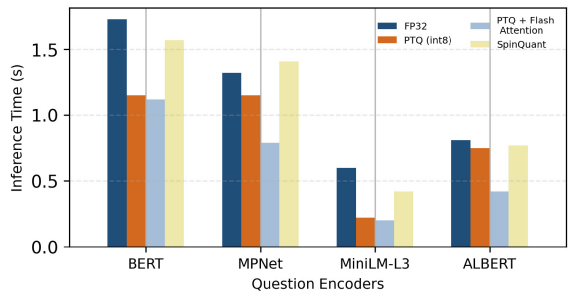

FIGURE 10
Question encoder inference time.

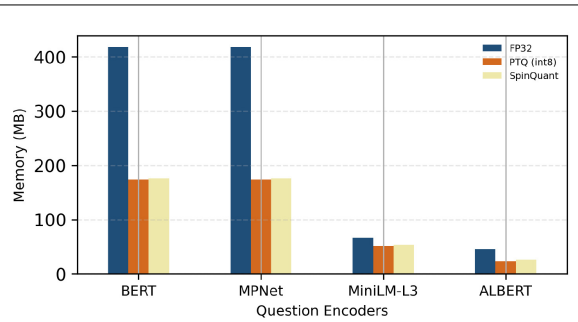

FIGURE 11
Question encoder memory footprint.

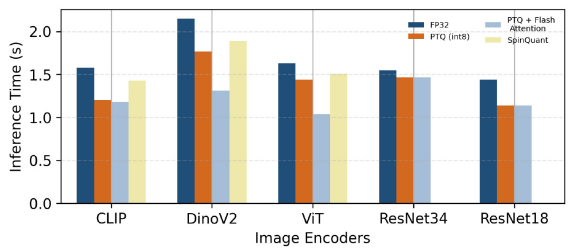

FIGURE 12
Image encoder inference time.

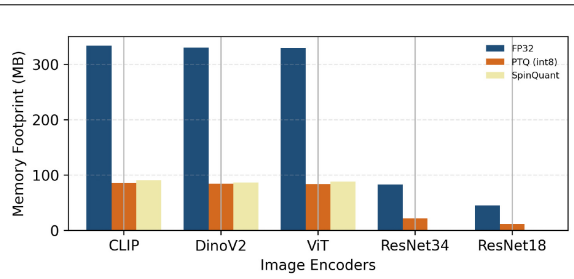

FIGURE 13
Image encoder memory footprint.

5.6 TinyKGI edge inference

To further validate efficiency gains on real edge hardware, we performed inference experiments on a Jetson Nano 2GB Developer Kit to demonstrate efficiency gains on a resource-constrained device while maintaining high accuracy. Since INT8 inference is not natively supported on the Jetson Nano GPU, all experiments were conducted on the CPU.

We evaluated our best TinyKGI configurations, comparing a full-precision FP32 model using MpNet with a quantized INT8 model using MiniLM-L3 question features. Inference was performed using ONNX Runtime for both FP32 and INT8 models due to limitations of the PyTorch quantization backend on the Jetson Nano.

Table 7 presents the F1-scores for FP32 and INT8 TinyKGI model inference on Jetson Nano for all the knowledge gaps in the GQA dataset. The FP32 model achieves a Macro-F1 score of 99.11%, while the INT8 model achieves 99.0%, indicating minimal performance degradation. In terms of memory footprint (for the TinyKGI classifier), the FP32 model requires 1325.91 KB, whereas the INT8 model requires 337.23 KB, corresponding to approximately 3.9× memory reduction. For inference time, the FP32 model takes 6.105 ms, while the INT8 model takes 3.645 ms for classifier-only inference, resulting in approximately 1.7× speedup per batch. These results demonstrate that TinyKGI achieves real efficiency gains on edge devices with negligible impact on accuracy.

6 Discussion

For question-only Knowledge Gap Identification (TinyKGI-Q), transformer-based encoders consistently outperform prior work, with MiniLM-L3 emerging as the best trade-off between accuracy and efficiency. Quantization results show that post-training quantization (PTQ), particularly when combined with Flash Attention, provides substantial inference speedups while preserving accuracy. SpinQuant leads to decrease in memory footprint, yields comparatively smaller performance gains for the KGI task, suggesting that Knowledge Gap Identification does not strongly benefit from rotation-based quantization strategies.

In contrast, incorporating image features (TinyKGI-Q+I) does not improve Knowledge Gap Identification performance and often results in a slight decrease in Macro-F1 scores. This observation highlights that KGI primarily depends on linguistic and question features rather than visual content, distinguishing it from VQA tasks. However, the multi-modal model inference using TinyML optimizations, enables flexible deployment on edge when visual context is required.

Figure 1 presents examples from the three VQA datasets, highlighting salient phrases and their corresponding Knowledge Gap tags. From these examples, we observe that Knowledge Gaps are highly dependent on the question itself and can be effectively predicted using question features alone, which further explains why incorporating visual features does not provide additional benefits in this setting.

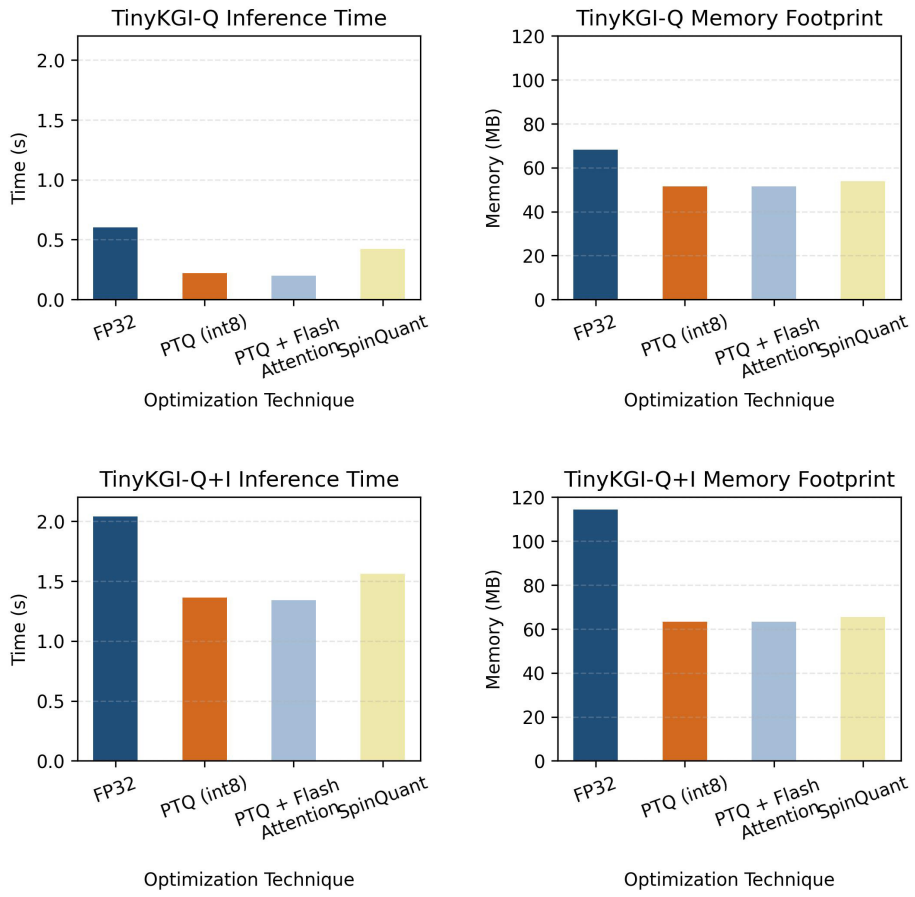

FIGURE 14
TinyKGI end-to-end inference time and memory footprint across optimization techniques.

TABLE 7 TinyKGI edge inference F1-scores for best full-precision and quantized question-only models for GQA dataset.

Knowledge Gap	TinyKGI (Mpnet-Base) (fp32)	TinyKGI (MiniLM-L3) (int8 + flash attention)
Activity	99.4	99.4
Attribute	99.6	99.5
Direction	98.8	98.8
Entity resolution	99.5	99.4
Location	99.1	99.2
Material	98.3	97.9
Reasoning	99.7	99.5
Sentiment	99.8	99.8
Size	99.3	99.3
State	97.6	97.7
Macro F1	**99.11**	**99.0**

7 Conclusion

In this work, we introduced Tiny Knowledge Gap Identification (TinyKGI), a lightweight and efficient framework for identifying Knowledge Gaps in multi-modal reasoning systems. Inspired by human cognitive processes, TinyKGI leverages cognitive skill mappings to predict plausible Knowledge Gap tags that may lead to erroneous reasoning in AI systems. Through Knowledge Gap Identification, we obtain interpretable insights into model behavior while enabling efficient inference in resource-constrained environments.

We evaluated TinyKGI across three Visual Question Answering datasets and demonstrated consistent state-of-the-art Macro-F1 performance, with minimal accuracy drop under low-precision inference. From an efficiency perspective, TinyKGI achieves substantial reductions in inference time and memory footprint through TinyML optimization techniques, particularly post-training quantization combined with Flash Attention. End-to-end evaluation shows that TinyKGI enables fast and memory-efficient inference, making it well suited for on-device and edge AI applications. Overall, TinyKGI provides a robust and scalable framework for understanding and improving reasoning in AI systems, with direct implications for human–AI teaming and edge deployment.

In this work, we focus on the identification of knowledge gaps for a given task. Once identified, these gaps can be resolved through various strategies, including human-in-the-loop interaction and external knowledge retrieval. While knowledge gap identification enables the agent to identify the reasoning skills required for a given task and is challenging, integrating these predictions into downstream applications such as improving VQA performance

or human–AI interaction remains an important direction. Such resolution and downstream integration are beyond the current scope of this paper and can be extended for future work.

Data availability statement

The Knowledge Gap (KG) annotations for the GQA and CLEVR datasets are publicly available through the following repositories: Hugging Face: https://huggingface.co/datasets/Sarikaa-Sridhar/tinykgi-kg-annotations, GitHub: https://github.com/Sarikaa3/TinyKGI-KG-Annotations. For the TDIUC dataset, the original question types were used as Knowledge Gap labels and therefore no additional annotation files were released. The original datasets used in this study are publicly available: TDIUC at https://kushalkafle.com/projects/tdiuc.html#download, GQA at https://cs.stanford.edu/people/dorarad/gqa/download.html, and CLEVR at https://cs.stanford.edu/people/jcjohns/clevr/. Further inquiries can be directed to sridhar.86@buckeyemail.osu.edu.

Author contributions

SS: Data curation, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. SG: Data curation, Investigation, Methodology, Software, Validation, Writing – review & editing. GB: Data curation, Investigation, Software, Validation, Writing – review & editing. CM: Conceptualization, Funding acquisition, Investigation, Methodology, Resources, Supervision, Writing – review & editing. SP: Conceptualization, Funding acquisition, Investigation, Methodology, Resources, Supervision, Writing – review & editing.

Funding

The author(s) declared that financial support was received for this work and/or its publication. The authors acknowledge support from the National Science Foundation (NSF) grant #2112471 (AI-EDGE). Any opinions and findings are those of the author(s) and do not necessarily reflect the views of the granting agency.

Acknowledgments

The authors also thank Prof. Hari Subramoni for providing the Jetson Nano device used for the edge-device evaluation conducted in this study.

Conflict of interest

GB was employed by Amazon AGI.

The remaining author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2026.1817034/full#supplementary-material

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., et al. (2024). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., et al. (2015). VQA: visual question answering. *CoRR, abs/1505.00468*. doi: 10.1109/ICCV.2015.279

Bajaj, G., Bandyopadhyay, B., Schmidt, D., Maneriker, P., Myers, C., and Parthasarathy, S. (2020). "Understanding knowledge gaps in visual question answering: Implications for gap identification and testing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 386–387. doi: 10.1109/CVPRW50498.2020.00201

Bajaj, G., Current, S., Schmidt, D., Bandyopadhyay, B., Myers, C. W., and Parthasarathy, S. (2022). Knowledge gaps: a challenge for agent-based automatic task completion. *Topics Cogn. Sci.* 14, 780–799. doi: 10.1111/tops.12584

Bajaj, G. K. (2024). *Detection, Identification, and Resolution of Knowledge Gaps in Visual Question Answering Agents*. Columbus: The Ohio State University.

Ball, J., Myers, C., Heiberg, A., Cooke, N. J., Matessa, M., Freiman, M., et al. (2010). The synthetic teammate project. *Computat. Mathem. Organ. Theory* 16, 271–299. doi: 10.1007/s10588-010-9065-3

Carroll, J. B. (1993). *Human Cognitive Abilities: A Survey of Factor-Analytic Studies*. Number 1. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511571312

Collins, A., Warnock, E. H., Aiello, N., and Miller, M. L. (1975). "Reasoning from incomplete knowledge," in *Representation and Understanding* (Elsevier), 383–415. doi: 10.1016/B978-0-12-108550-6.50018-5

Dao, T. (2023). "Flashattention-2: faster attention with better parallelism and work partitioning," in *International Conference on Learning Representations*, 35549–35562.

Dosovitskiy, A. (2020). An image is worth 16x16 words: transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Gentner, D., and Collins, A. (1981). Studies of inference from lack of knowledge. *Memory Cogn*. 9, 434–443. doi: 10.3758/BF03197569

Gonzalez, C., and Dutt, V. (2011). Instance-based learning: integrating sampling and repeated decisions from experience. *Psychol. Rev*. 118:523. doi: 10.1037/a0024558

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6904–6913. doi: 10.1109/CVPR.2017.670

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. doi: 10.1109/CVPR.2016.90

Hudson, D. A., and Manning, C. D. (2019). GQA: a new dataset for compositional question answering over real-world images. *CoRR, abs/1902.09506*.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., et al. (2018). "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2704–2713. doi: 10.1109/CVPR.2018.00286

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. (2017). "Clevr: a diagnostic dataset for compositional language and elementary visual reasoning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1988–1997. doi: 10.1109/CVPR.2017.215

Kafle, K., and Kanan, C. (2017). "An analysis of visual question answering algorithms," in *Proceedings of the IEEE International Conference on Computer Vision*, 1965–1973. doi: 10.1109/ICCV.2017.217

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., et al. (2023). "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4015–4026. doi: 10.1109/ICCV51070.2023.00371

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behav. Brain Sci*. 40:e253. doi: 10.1017/S0140525X16001837

Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., Gan, C., and Han, S. (2022). "On-device training under 256kb memory," in *Advances in Neural Information Processing Systems*, 22941–22954. doi: 10.52202/068431-1667

Liu, M., Chen, C., and Gurari, D. (2024). An evaluation of gpt-4v and gemini in online vqa. *arXiv preprint arXiv:2312.10637*.

Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., et al. (2024). Spinquant: LLM quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

Marino, K., Rastegari, M., Farhadi, A., and Mottaghi, R. (2019). "Ok-vqa: a visual question answering benchmark requiring external knowledge," in *Proceedings of the IEEE/cvf Conference on Computer Vision and Pattern Recognition*, 3195–3204. doi: 10.1109/CVPR.2019.00331

Motlagh, N. K., Davis, J., Anderson, T., and Gwinnup, J. (2022). Learning when to say" i don't know". *arXiv preprint arXiv:2209.04944*.

Newell, A., Simon, H. A., et al. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-hall.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., et al. (2023). Dinov2: learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Özdemir, Ö., and Akagündüz, E. (2024). "Enhancing visual question answering through question-driven image captions as prompts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1562–1571. doi: 10.1109/CVPRW63382.2024.00163

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., et al. (2021). "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning* (PmLR), 8748–8763.

Reimers, N., and Gurevych, I. (2019). "Sentence-bert: sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics). doi: 10.18653/v1/D19-1410

Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., et al. (2024). Grounded sam: assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*.

Schmidt, D. P. (2020). *Identifying knowledge gaps using a graph-based knowledge representation*. Master's thesis, Wright State University.

Spelke, E. S. (2017). Core knowledge, language, and number. *Lang. Learn. Dev*. 13, 147–170. doi: 10.1080/15475441.2016.1263572

Team, G., Anil, R., Borgeaud, S., Alayrac, J. B., Yu, J., Soricut, R., et al. (2024). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Zhang, X., He, J., Zhao, J., Hu, Z., Yang, X., Li, J., et al. (2024). Exploring and exploiting model uncertainty for robust visual question answering. *Multim. Syst*. 30:348. doi: 10.1007/s00530-024-01560-0