# Towards Sustainable Knowledge Gap Identification with Tiny Machine Learning Techniques

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Sarikaa Sridhar

Graduate Program in Computer Science and Engineering

The Ohio State University

2024

Thesis Committee

Srinivasan Parthasarathy, Advisor

John Paparrizos

# Abstract

Identifying the lack of cognitive capabilities in artificially intelligent systems has been a growing field and a necessary step. Knowledge gaps (KG) are a lack of insufficient information which may lead to poor cognitive capabilities. Knowledge gap identification can help predict where intelligent systems go wrong. This work proposes methods to identify knowledge gaps in Visual Question Answering (VQA) datasets. We created a model to automatically classify questions and image pairs into different knowledge gap categories that can later be used to resolve shortcomings of VQA models. Additionally, artificially intelligent systems often require several days to train for the system to learn complex features to provide the most accurate predictions. Testing or inferencing with trained models also requires huge amounts of energy and emits massive amounts of $CO_2$. Thus, this work also aims to train a classification model which is the Knowledge Gap Identification (KGI) model in resource-constrained environments using TinyML (Tiny Machine Learning) techniques proposed by previous research. The two main techniques implemented are: Quantization-aware scaling and Sparse Update. Finally, this work aims to compare the original model with its tiny version (Sustainable KGI) using accuracy, processing time, energy consumed and estimation of carbon emission as evaluation metrics.

# Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Srinivasan Parthasarathy and Dr. Christopher Myers from AFRL, for consistent guidance and support at every stage of the project. Additionally, I wish to thank Goonmeet Bajaj, another PhD student under the supervision of Dr. Parthasarathy, who offered support and constructive feedback throughout the project's duration. Lastly, I extend my heartfelt appreciation to my parents for their unwavering support, belief in me, and encouragement to continually strive for my growth.

# Vita

August 2018 - May 2022 ..... B.E., Computer Science and Engineering, College of Engineering Guindy, Anna University

August 2022 - May 2024 ..... M.S., Computer Science and Engineering, The Ohio State University

May 2023 - July 2023 .....Research In Sustainable Energy (RISE) Internship, The Ohio State University

August 2023 - present .....Graduate Research Associate, The Ohio State University

Starting August 2024 .....PhD Computer Science and Engineering, The Ohio State University

# Publications

Sarikaa, S., and S. Niranjana. "Time series forecasting of cloud resource usage." *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2021.

# Fields of Study

Major Field:  Computer Science and Engineering

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1. Introduction

## 1.1 Problem Statement

Autonomous AI systems are more widespread today, which are built to accomplish a task, interact with the surroundings, and perform analysis. However, there are limitations in these systems such as: it is sensitive to slight changes in the input, and it is hard for a trained AI system to adapt to new tasks and environments. AI systems are trained to always provide an answer even if it's incorrect. This becomes very crucial in environments where there is human and AI teaming. For example, in Figure 1 considers an autonomous pilot tasked to navigate a drone. Hence building a robust AI system to prevent harmful situations is very important.  Thus, this work aims to build human cognitive abilities into AI systems.

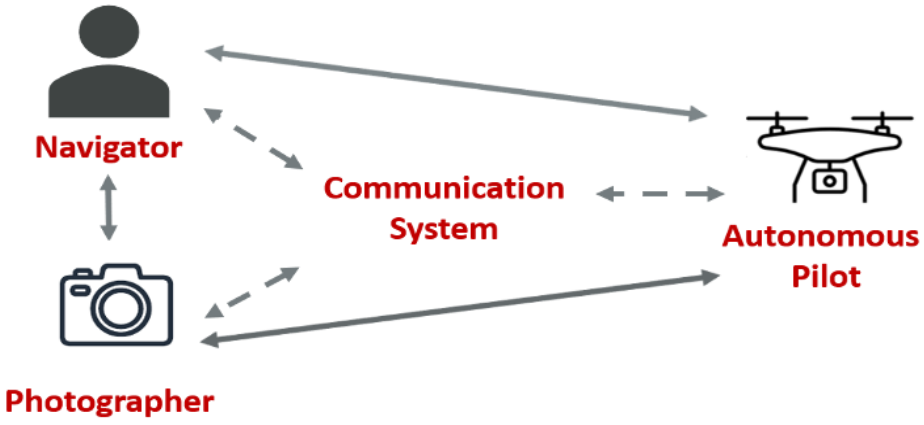



Figure 1Autonomous Artificial Intelligence System

When comparing human cognition and artificial intelligence systems, humans have the cognitive ability to find shortcomings in their knowledge and can take measures to identify and solve their knowledge gaps. In contrast, artificially intelligent systems require this

ability to be built to identify shortcomings in their reasoning skills and knowledge. To study this problem, researchers have looked into exploring Visual Question Answering (VQA) as a testbed [2]. VQA [1] requires reasoning abilities to understand images and natural language questions to generate the most accurate answer. VQA agents are said to have a Knowledge Gap (KG) when they predict an answer incorrectly due to incorrect reasoning over the image and question. These gaps in knowledge can lead to poor performance and decreased robustness.

Additionally, Artificial Intelligence systems require a significant amount of processing time, energy consumption, and carbon footprint in order to be effective in any task. Deep learning models, in particular, often require several days to finish training and for the model to learn complex features and provide the most accurate predictions. Testing or performing inference with trained models also requires huge amounts of energy and emits massive amounts of carbon-dioxide. Thus, this work also aims to address these issues and scale down the deep learning models for knowledge gap identification (KGI) task by implementing Tiny Machine Learning (TinyML) techniques [7] proposed by previous research. To decrease the training and inference times for this problem, these techniques entail several modifications to computing techniques and comparison is made between the original model (KGI) and its tiny version (Sustainable KGI).

The major research question investigated and addressed in this work is: *Can we build human cognitive capabilities into Artificially Intelligent systems with less computation and preserving accuracy?*

## 1.2 Contributions of this work

To address the research question, implementation is done in two parts as contribution:

(1) This work aims to build a multi-modal classification approach for fusing both image and question features to perform knowledge gap identification. For predicting knowledge gaps using question features only and the combination of question-image features as inputs, a Multi - Layer Perceptron [12] model is developed as a classifier. A comparison is made between all the KGI models and different fusion techniques.

(2) Secondly, for the sustainable KGI model, the objective is to reduce training time of the model, involving implementation of two methods Quantization-aware Training and Sparse Update. Finally, performance metrics of KGI and sustainable KGI models are compared.

The rest of the thesis is structured as follows. Chapter 2 describes related work, terminologies used and motivation. Chapter 3 provides a detailed explanation on the Knowledge Gap Identification model, the datasets used, extracting features from questions and images, and model architecture. Chapter 4 describes the TinyML methods implemented to build the Sustainable KGI model. Chapter 5 discusses results. Chapters 6 and 7 describe Challenges and Future Work, respectively.

# Chapter 2. Related work

## 2.1 Knowledge Gaps and Taxonomy

Autonomous AI systems are said to have knowledge gaps [8] when the system behaves flawed or predicts incorrect answers due to insufficient or lack of information or poor reasoning abilities. Hence detecting, identifying and resolving knowledge gaps or cognitive capabilities of the AI model is important. A taxonomy for knowledge gaps has been created and used for identifying knowledge gap tags for each question [2, 8]. Bajaj et al (2020) [2] discusses a refined version of the KG taxonomy particularly for VQA tasks. Figure 2 shows the taxonomy for knowledge gaps. Each KG denotes the reasoning abilities that the model needs, to provide the most accurate answer. This article also annotates each question with one or more KGs and performs more analysis on that. Further, Bajaj et al (2022) [3] provides a more detailed explanation on how knowledge gaps relate to cognitive capabilities and can be used in the AI systems to make it robust and flexible.

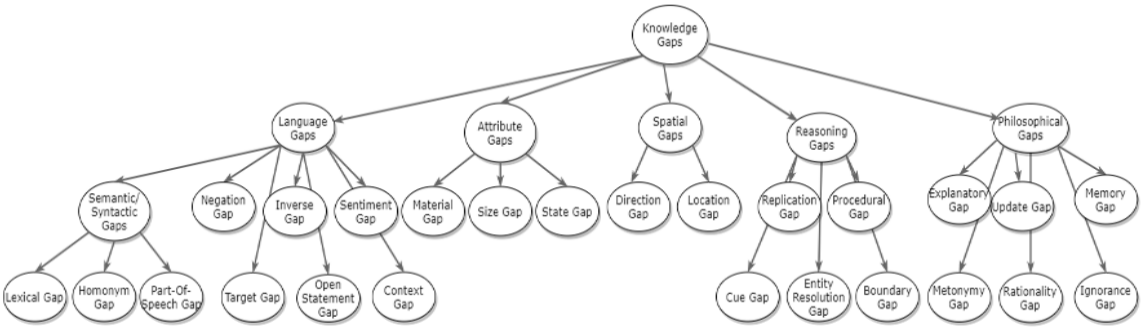



Figure 2 Knowledge Gap Taxonomy

## 2.2 Visual Question Answering

The reason for investigating KGs using visual question answering is, it is the intersection of language, vision, and reasoning. A VQA task can be defined as an algorithm which predicts answers for a question asked from the image. Hence the VQA model extracts the image feature embeddings and question feature embeddings and trains an AI model to generate the answer. Figure 3 provides the model architecture for the VQA model. Antol et al. (2015) [1] proposes the working of VQA, in detailed description about the data sets used and implements two methods, one using a multi-layer perceptron and another using LSTM [13] for generating the answer. Figure 4 provides two examples of a question image pair mapped to the corresponding knowledge gaps. In the first example, for the question "Where is the man?" the KG tags are "Entity resolution" and "Direction". Entity resolution gap is to identify if the man is present or not and the Direction gap to know which position in space the man is present. These two gaps fall under "Spatial, Perception, Attention" cognitive capabilities.

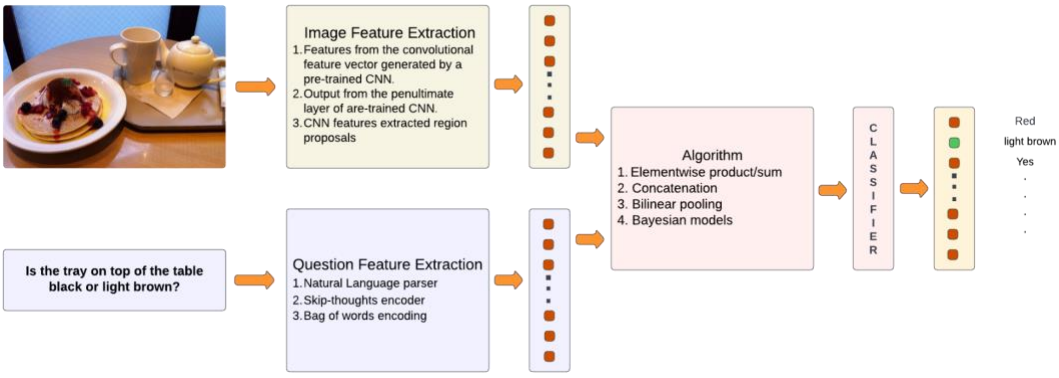


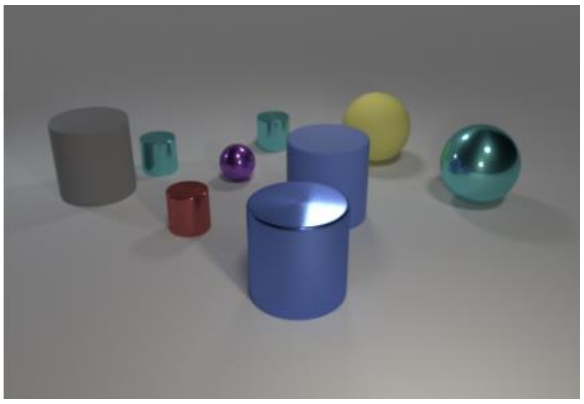

Figure 3 Model architecture of a VQA model

*__GQA Q__: *Where is the man?*

__GQA KG Tags__: Entity Resolution, Direction

*__TDIUC Q__: *What is the weather?*

__TDIUC KG Tags__: Scene Recognition



*__CLEVR Q:__ *There is a large object that is on the left side of the large blue cylinder in front of the rubber cylinder on the right side of the purple shiny thing; what is its shape?*

__CLEVR KG Tags:__ Entity Resolution, Direction, Attribute, Size

**GQA, TDIUC and CLEVR are two different VQA datasets*

Figure 4 Examples of question-image pair from two VQA datasets GQA and TDIUC with the corresponding KG labels

## 2.3 Knowledge Gap Identification

The function of the KGI method is to process the natural language question, image, and the knowledge gaps to predict the KGs for new question-image pairs. By identifying the type of KGs, the gaps can then be resolved to improve the prediction accuracy of the VQA models. In previous work, the metadata present in the VQA dataset was used to identify KGs. The tags are assigned by using the question's structural annotation as well as functional programs provided in the dataset, like yes/no type of question or query type

questions. For example, if the question type was ‘positionVerify’, ‘Direction Gap’ is assigned as it represents identifying spatial position [2]. A KG-VQA [3] paradigm has been defined to evaluate the cognitive capabilities of intelligent systems by identifying the knowledge gaps for different Visual Question Answering datasets. A Bidirectional Encoder Representation from Transformers (BERT) [14] model is fine-tuned for VQA questions and used as a classifier for predicting the knowledge gaps [3].

The prior research in identifying possible knowledge gaps for VQA questions is based on metadata associated with questions and uses rule-based methods for classifying. However, existing methods cannot be used for datasets that do not have a rich set of metadata associated with the questions (i.e., VQA 2.0 dataset) [2]. Therefore, this work aims to define a generic method for tagging knowledge gaps to VQA questions for different datasets that do not have such metadata. For automatically identifying knowledge gaps, a classifier model is built that uses both the image and the question features to learn the knowledge gap tags.

## 2.4 Tiny Machine Learning

The Tiny Machine Learning paradigm [7, 9, 10, 11] aims to implement methods to increase the model performance by less computation, less data for training, and inferencing. Mainly developed to build smaller models and deploy those models on edge devices to perform inferencing on low power hardware. Lin, Ji, 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 inferencing after code generation. Hence training and testing AI models using TinyML techniques will result in a smaller carbon footprint promoting Green AI. Thus, this work aims to extend TinyML methods to the knowledge gap identification task to develop sustainable methods for identifying knowledge gaps.

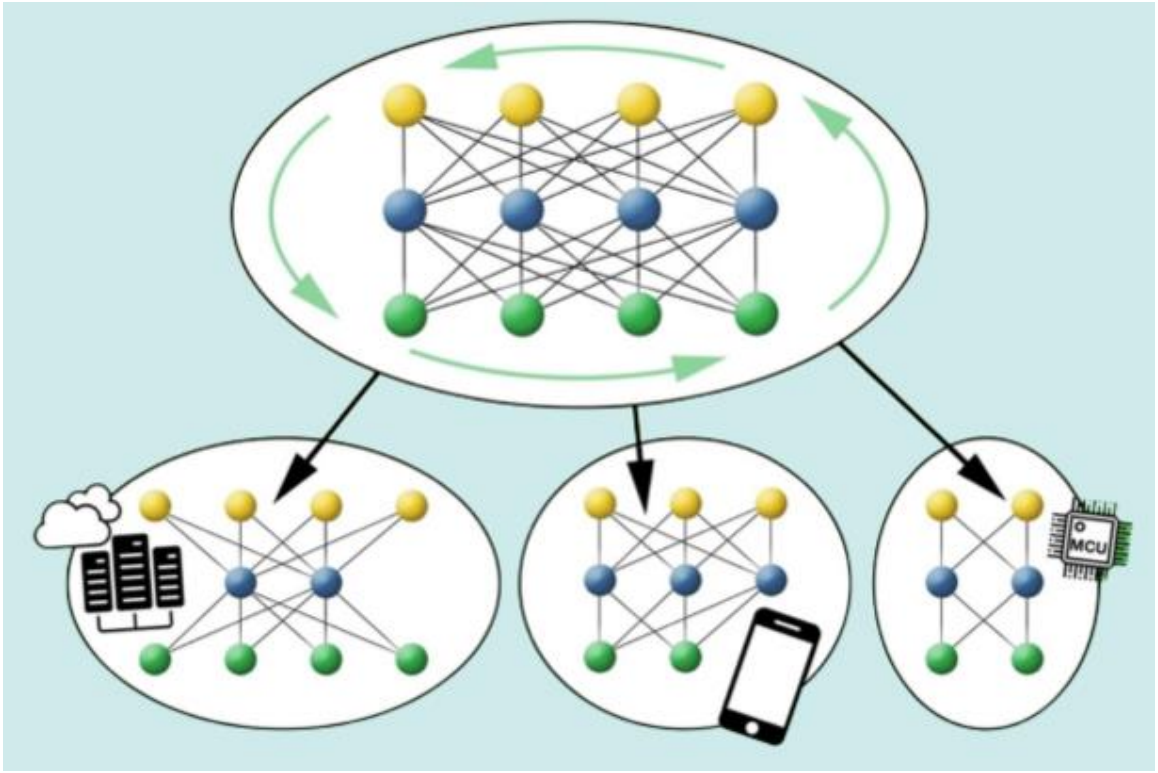



Figure 5 Applications of TinyML

# Chapter 3. Knowledge Gap Identification

## 3.1 VQA Datasets

In this section, all the VQA datasets used are discussed. The knowledge gap identification dataset contains the questions, images, and KG tags from [2]. The author has used rule-based mapping for KGs to question-image pairs. The dataset for which the mappings are applied and KGI is performed are: GQA [4], TDIUC [6] and CLEVR [5]. Table 1 shows the three datasets used and the KGs categorized based on the metadata present in the dataset. Table 2 shows the number of datapoints used for training, validation and testing the KGI model.

| VQA Datasets | Knowledge Gaps |
|---|---|
| **GQA** | Activity, Attribute, Direction, Entity Resolution, Location, Material, Reasoning, Sentiment, Size, State |
| **TDIUC** | Object Presence, Color, Scene Recognition, Counting, Attribute, Activity Recognition, Positional Reasoning, Sport Recognition, Object Recognition, Utility Affordance, Sentiment Understanding |
| **CLEVR** | Attribute, Counting, Direction, Entity Resolution, Material, Size |

Table 1 VQA Datasets and its corresponding KG tags

The GQA [4] dataset was developed for real-world visual reasoning and consists of real-world questions and images. A question engine which utilizes the information from scene graphs, attributes and objects is used to generate a diverse set of 22 million reasoning questions with the functional programs. The semantic representations associated with questions are used to decrease bias in the dataset and it is down sampled to a balanced

dataset, also for each question the distribution of answers is made uniform. The semantic representation extracted from functional programs is used to assign KGs to the questions [2] which is used as a supervised dataset for the KGI model. Here, each question-image belongs to one or more KGs. Figure 6 shows the distribution of questions per knowledge gap.

Task Directed Image Understanding Challenge (TDIUC) [6] dataset contains 1.6 million questions. Based on the task the questions solve, the questions are categorized into different types which are used as reasoning abilities needed for the model to provide an answer. The question types are listed as the KGs shown in Table 1. For this dataset, each question-image pair belongs to only one knowledge gap.

The Compositional Language and Elementary Visual Reasoning (CLEVR) [5] is a diagnostic dataset with minimum bias in data and covers a range of reasoning abilities. The images in this dataset are composed of 3D objects for simplifying recognition. The knowledge gaps are identified using the functional programs representing the questions similar to the GQA dataset. In CLEVR, one or more KGs are assigned to each question-image pair.

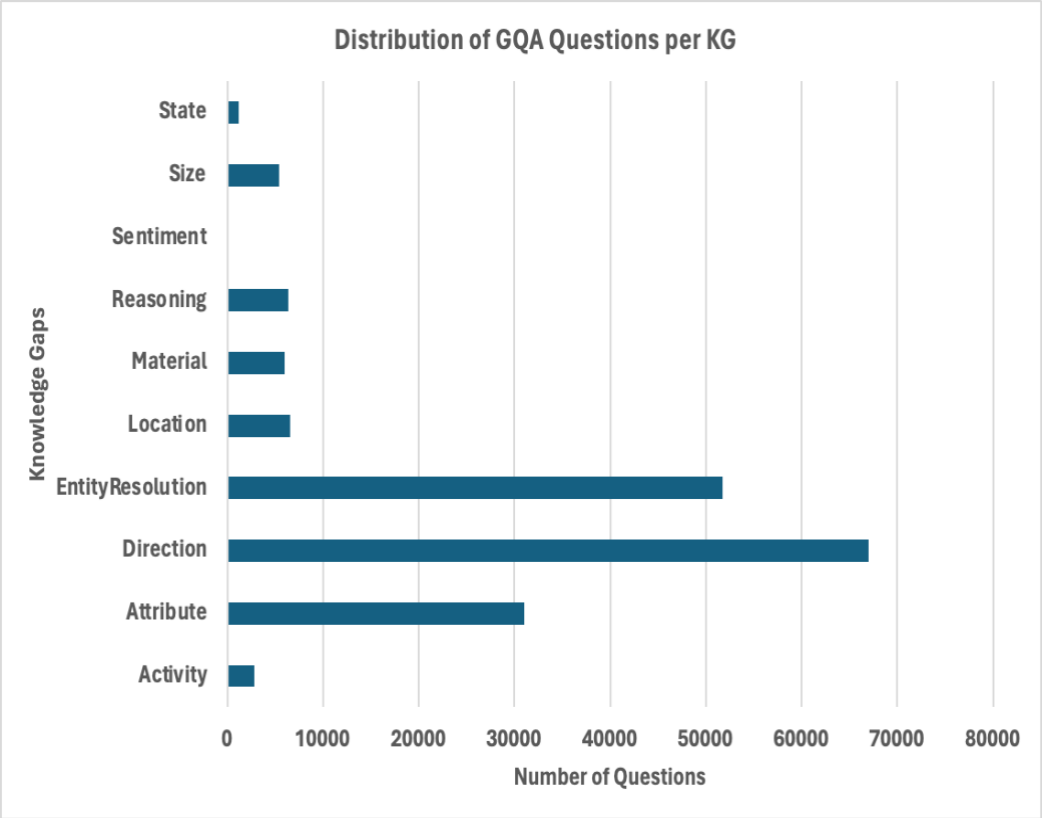



Figure 6 Distribution of GQA Questions per KG

| VQA Dataset | Train Question-Image samples | Validation Question-Image samples | Test Question-Image samples |
|---|---|---|---|
| **GQA** | 921818 | 184364 | 117286 |
| **TDIUC** | 695244 | 173812 | 418457 |
| **CLEVR** | 559991 | 139998 | 149991 |

Table 2 Train, Validation and Test set split for three datasets

## 3.2 Feature extraction

Bidirectional Encoders Representations from Transformers (BERT) [14] is used for extracting question feature vectors. It is a bidirectionally trained language model that can learn contextual relationships between words in the given input text. BERT uses Masked LM and Next Sentence Prediction networks for learning the context. First, BERT tokenizer is used to tokenize each question text and all the question tokens are padded to a maximum length. The padded tensors containing tokens are given as input to the BERT pretrained model to get the output predictions. This model has a total of 768 hidden units, out of which the first position, also called [CLS] token is sliced from the output tensor. The reason for using only [CLS] tokens is, during model training sentence-wide sense is encapsulated in the first position of the output prediction. Hence dimension for each question feature vector is (768,).

The feature embeddings for images given the datasets are used. For GQA, faster-RCNN and ResNet - 101 models were used for extracting object and spatial image features respectively. For TDIUC, ResNet - 152 is used and Convolutional Neural Network (CNN) [15] is used for CLEVR. These image features are given as input to a CNN to learn the complex features and reduce the feature vector dimension for downstream tasks. The CNN architecture consists of a Convolution layer, followed by a Maxpool layer, then ReLU activation function, Batch normalization and a final Convolution layer with varying hidden units tuned depending on the dimensions of the image features.

## 3.3 Combining Image and Question Features

Two methods are implemented for fusing the image and question features together for down-stream classification: Dot Product and Concatenation.

### 3.3.1 Dot product

The matrix product of learned image features and question embeddings are computed. Let A be a $b \times n_1 \times n_n$ dimension tensor and B is $b \times n_n \times n_m$ dimension tensor, where b is the batch size.

D = A . B is of dimension $b \times n_1 \times n_m$ , where (.) represents dot product.

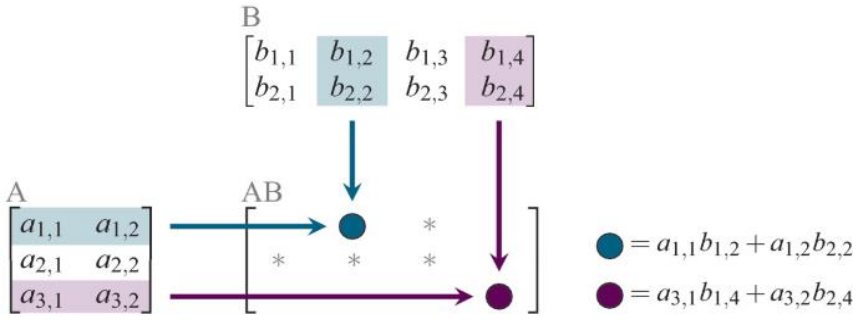



Figure 7 Illustration for Dot Product

### 3.3.2 Concatenation

The learned image features from CNN and BERT question embeddings are concatenated in the specified dimension of the tensor. Let A be a $n_1 \times n_n \times \ldots \times n_i \times \ldots \times n_n$ dimension tensor, and B be a$n_1 \times n_n \times \ldots \times n'_i \times \ldots \times n_n$ dimension tensor. The concatenation of A and B horizontally along dimension i results in a new tensor C = [A B] of size

$n_1 \times n_n \times \ldots \times (n_i + n'_i) \times \ldots \times n_m$, where $n'_i$ represents the size of dimension ( i ) of tensor B. Here, [] represents concatenation operation.

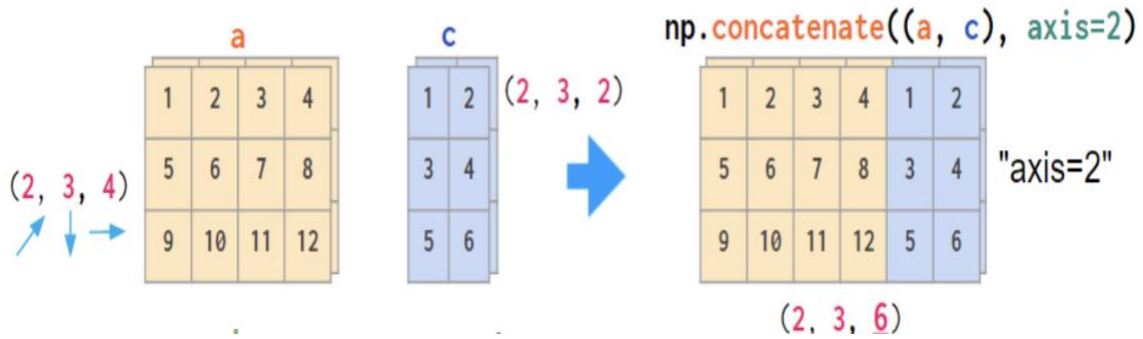



Figure 8 Illustration for Concatenation

## 3.4 Multi - Layer Perceptron (MLP) Classifier

As a final classifier Multi-Layer Perceptron neural network is used. The image and question features combined with the knowledge gap tags are given as the input to the network. The knowledge gap tags are encoded using a multi-label binarizer. This network learns the features and predicts the KGs for new question-image pairs. Multi-label or multi-class classifiers of knowledge gaps for image-question pairs in VQA datasets can be built, which can automatically tag knowledge gaps for datasets without much metadata. The GQA and CLEVR datasets have one or more KGs for each question hence multi-label classification is performed. For TDIUC dataset, only one KG tag for each question so multi-class classification is performed. The model architecture of the MLP is a Linear layer, followed by ReLU activation, Batch Normalization, Linear layer, TanH activation and then final Linear layer. For the output activation, sigmoid is used for multi-label and softmax is used for multi-class. For GQA and CLEVR after applying sigmoid, the output

values with threshold greater than 0.5 are taken as positive prediction. For TDIUC, maximum of the outputs from softmax function is taken as positive predicted KG. Figure 6 provides a block diagram for the knowledge gap identification model illustrated with an example. For loss function, binary cross entropy loss is used.

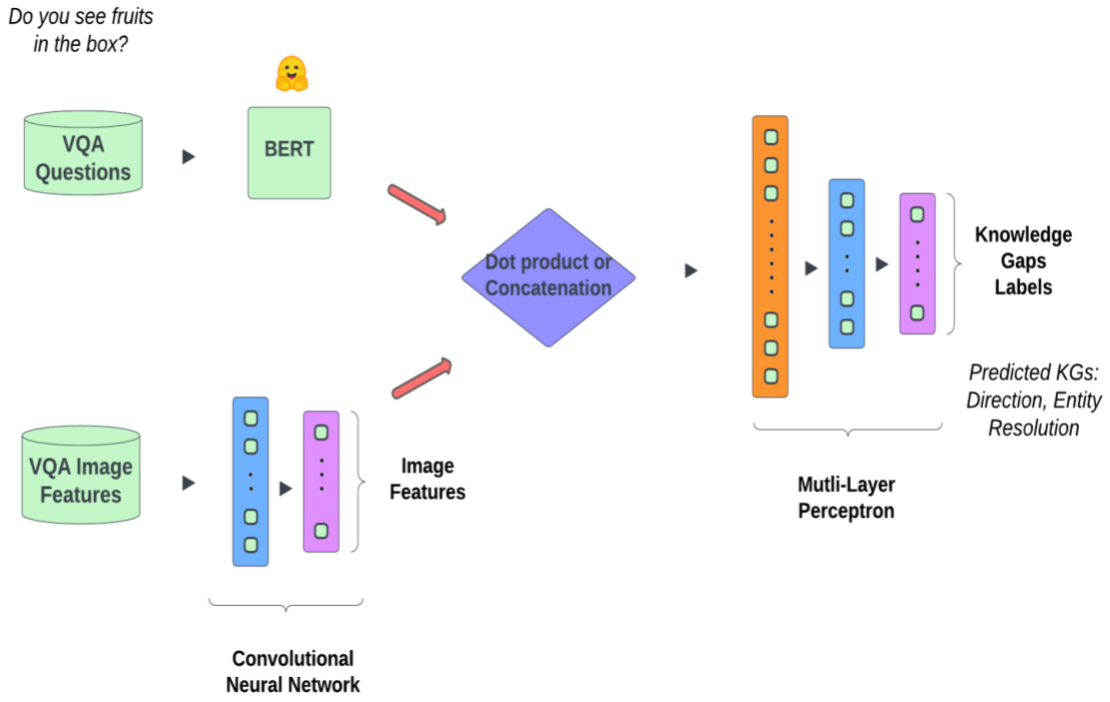



Figure 9 KGI Model Architecture

# Chapter 4. Sustainable Knowledge Gap Identification

The pre-trained KGI model is taken and TinyML techniques are applied for fine-tuning tasks. The Sustainable KGI model aims to improve the efficiency of the Multi-Layer Perceptron model by reducing computation which results in reduction of training time. Two methods are implemented: Quantization-aware scaling and Sparse Update [7].

## 4.1 Quantization - Aware Scaling (QAS)

Quantization-aware scaling [7, 16, 17] method involves converting the values from higher bit precision to lower bit precision by scaling. For a sustainable KGI model, a pre-trained KGI model is trained with 32-bit floating precision and is converted to 8-bit integer precision during fine-tuning the model. INT8 is faster for inference and is less expensive. Here, quantization-aware training is performed which reduces the bit precision of gradients, inputs, and weights of a model pre-trained with high precision and further fine-tunes by applying quantization.

For 32-bit floating point, $y_{fp32} = W_{fp32}\, x_{fp32} + b_{fp32}$ is the equation for the linear layer. The equation for quantized int8 is $y'_{int8} = convert\_to\_int(s_{fp32} \cdot (W'_{int8}\, x'_{int8} + b'_{int32}))$. The formula for updating the weight is $W''_{int8} = convert\_to\_int(W'_{int8} + \alpha \cdot G_{W'})$. $(\cdot\,')$ denotes the quantized tensor. $W$ is the weight tensor, $x$ is input, $b$ is bias, $y$ is output, $G$ is gradient and $\alpha$ represents the learning rate.

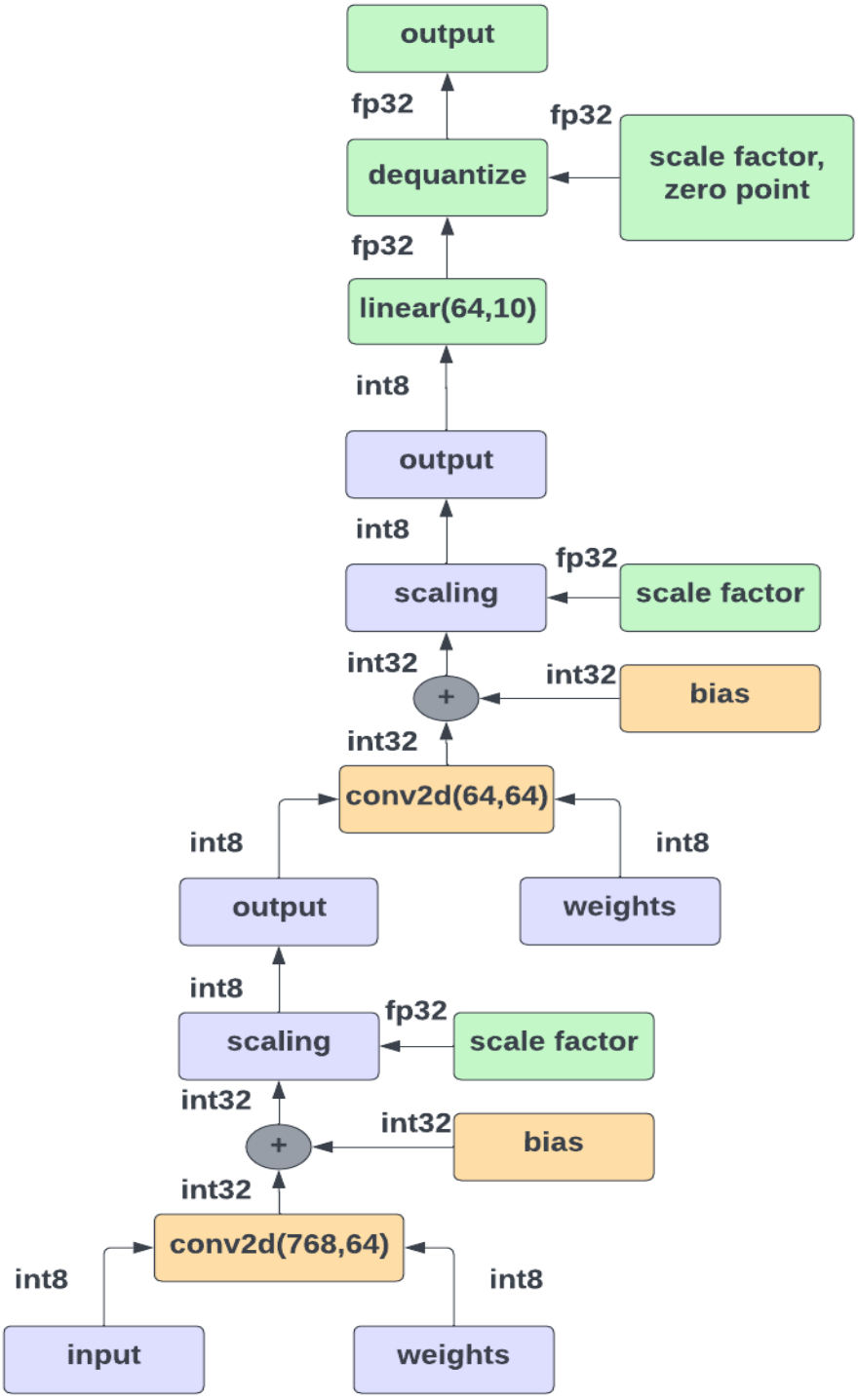



Figure 10 Sustainable KGI model architecture with QAS

### 4.1.1 Quantization

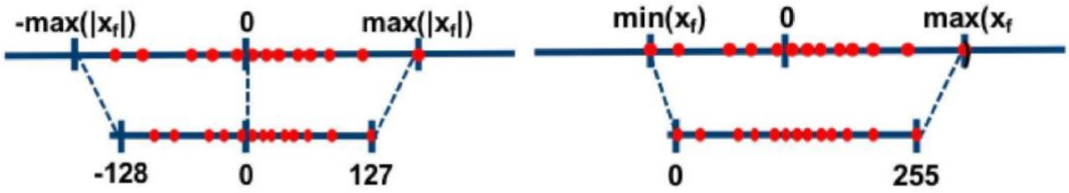
The aim is to map the values in the range $[\beta, \alpha]$ to be within $[-2^{b-1}, 2^{b-1} - 1]$, $b$ is the number of bits. The two main variables used for quantization are: scaling factor and zero point. To quantize a vector from the range of floating-point numbers to the quantized range, in this case $[\beta, \alpha]$ to [-128 to 127], the scaling factor is the ratio of range of floating point and quantized range. Zero point is calculated to represent the 0.0 of floating point in the quantized range. When zero point is non-zero it is affine quantization and if it is zero then it is symmetric quantization (Figure 9a). For sustainable KGI, for inputs and outputs, affine quantization (Figure 9b) is used, and symmetric quantization is used for weights and gradients.

$$x_{quantize} = clamp(round(x \div scale\_factor) + zero\_point)$$

$$\text{were, } scale\_factor = (\alpha - \beta) \div (2^{b-1})$$

$$zero\_point = -round(\beta * scale\_factor) - 2^{b-1}$$

Round function rounds a value to the nearest integer. The quantized value is then clamped between -128 to 127. $\alpha$, $\beta$ are the maximum and minimum value of the tensor.




(a) Symmetric quantization (b) Asymmetric quantization

Figure 11 Quantization Scheme Types

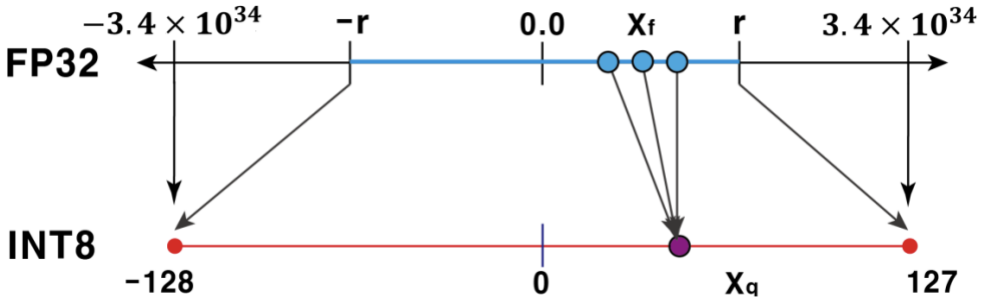



Figure 12 Quantization

### 4.1.2 Dequantization

To reverse the quantization.

$$x = (x_{quantize} - zero\_point) * scale\_factor$$

### 4.1.3 Quantizing weights and gradients

$W \epsilon R^{n \times m}$ represents the 2-dimensional weight matrix of a linear layer, where n, m are input and output channels. For per-tensor quantization, the scaling factor for $W$ is computed and denoted as $s_w \epsilon R$, such that the largest magnitude of the quantized matrix is $2^7$- 1 = 127

$$W \approx^{quantize} s_W \cdot W', G'_W \approx s_W \cdot G_W$$

The reason why quantization works is because what matters is not the absolute values of output by the model, but rather the relative difference between the probabilities it assigns to different classes for its predictions. Quantizing the weights only scales this relative difference up or down, without altering which class the network assigns the highest

probability to. As a result, it has no impact on the final predictions made by the neural network.

## 4.2 Sparse Update

Sparse Update [7] method involves weight update and gradient computation sparsely for layers or tensors. The gradients are pruned during backpropagation and the model is updated sparsely. There are three versions in the sparse update method.

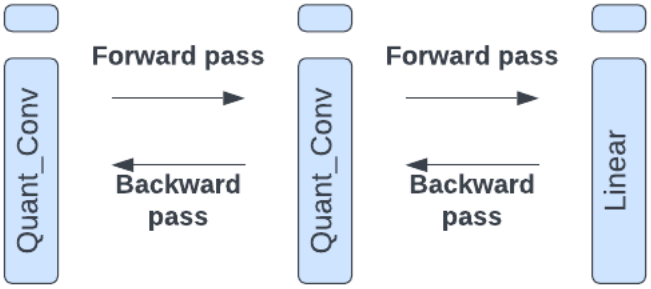



Figure 13 Full update

### 4.2.1 Sparse Layer Update

The weights of only a subset of layers are updated. In Figure 11, the pink shade represents 'fixed' and blue represents the layers that are updated. The small block on top of the layer block represents bias and the same applies for Figure 12 and 13.

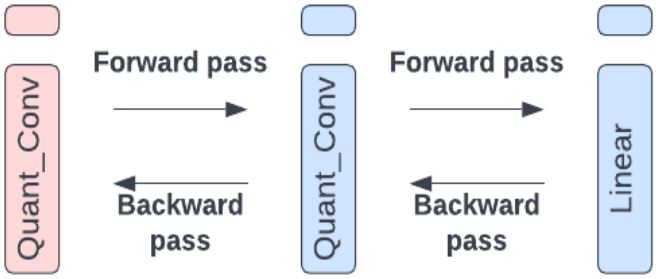



Figure 14 Sparse Layer Update

### 4.2.2 Bias Update

To update bias, this represents the number of layers to backpropagate to do so.

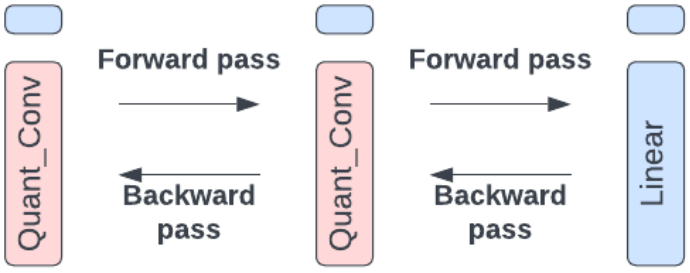



Figure 15 Bias Update

### 4.2.3 Sparse Tensor Update

Only the ratio of weight tensor is updated. To choose the layers for sparse update contribution analysis is used for selection. The contribution of each weight or bias towards the accuracy is computed. By maximizing the total contribution, the subset of layers and weights to be updated is found.

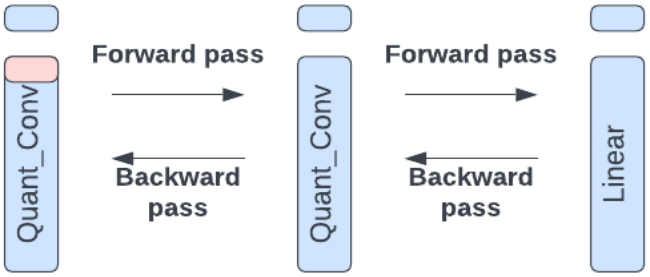



Figure 16 Sparse Tensor Update

## 4.3 Methodology

Convolutional Neural Networks (CNN) is trained as the classifier for predicting KGs in sustainable KGI. The question features and KGs are given as input to CNN and trained to predict the KGs. The scaling factor, zero points are computed and stored for the model training with 32-bit floating point precision. Using these variables the quantization is applied during fine-tuning the pre-trained KGI model. For fine-tuned sustainable KGI, quantized convolution and linear layers are used.

# Chapter 5. Results and Discussion

In this section, the results for KGI and sustainable KGI models are discussed. All the models are implemented using PyTorch library in python. The KGI model is trained with 0.0001 learning rate, 32 batch size and Adam optimizer for all three datasets. Table 3 shows the F1-scores for KGI model trained using question features only and question-image features combined. All the values in bold are the highest f1-scores. Except for the "sentiment" gap the KGI question features model performs better than model trained on combined features but there is no significant difference in the f1-score for question and combined features.

| **Knowledge Gap** | **F1-Score** | | |
|---|---|---|---|
| | **Question Feature** | **Question and Image Feature** | |
| | | **Dot Product** | **Concatenation** |
| **Activity** | **0.907** | 0.901 | 0.891 |
| **Attribute** | **0.961** | 0.957 | 0.952 |
| **Direction** | **0.956** | 0.955 | 0.954 |
| **Entity Resolution** | **0.976** | 0.971 | 0.974 |
| **Location** | **0.789** | 0.788 | 0.786 |
| **Material** | **0.839** | 0.803 | 0.805 |
| **Reasoning** | **0.968** | 0.965 | 0.963 |
| **Sentiment** | 0.792 | **0.816** | 0.758 |
| **Size** | **0.856** | 0.842 | 0.843 |
| **State** | **0.828** | 0.812 | 0.816 |

Table 3 F1-score for KGI model trained on GQA question features only, Dot product and concatenation of question and image features

Table 4 and Table 5 show the f1-scores of the KGI model trained for CLEVR and TDIUC datasets respectively.

| Knowledge Gaps | F1 - score |
|---|---|
| **Attribute** | 0.9997 |
| **Counting** | 1 |
| **Direction** | 1 |
| **Entity Resolution** | 1 |
| **Material** | 0.9994 |
| **Size** | 0.9998 |

Table 4 F1-score for KGI model trained on CLEVR question features

| Knowledge Gaps | F1 - score |
|---|---|
| **Object Presence** | 0.97 |
| **Color** | 0.96 |
| **Scene Recognition** | 0.95 |
| **Counting** | 0.99 |
| **Attribute** | 0.94 |
| **Activity Recognition** | 0.73 |
| **Positional Reasoning** | 0.93 |
| **Sport Recognition** | 0.92 |
| **Object Recognition** | 0.99 |
| **Utility affordance** | 0* |
| **Sentiment Understanding** | 0.84 |

Table 5 F1-score for KGI model trained on TDIUC question features

The KGI model is trained for 10 epochs and is fine tuned for different data points of the GQA. The sustainable KGI model is trained and fine-tuned for the GQA dataset. Here, the comparison is made between the model fine-tuned without applying TinyML methods (Quantization - Aware Scaling (QAS) and Sparse Update) and with applying the methods (Table 6). The model parameters are learning is 0.1, batch size is 64 and Adam optimizer is used. Sparse Update is applied for the two quantized convolutional layers with 0.75 weight update ratio for both the layers with bias of all layers updated. Here, for Direction Gap the performance decreases by 52% than the fine-tuned results, the reason for this is, this KG has the highest number of data and there are lot of false negative predictions, similarly for 'Attribute' and 'Entity Resolution'.

| Knowledge Gaps | KGI (without QAS + Sparse Update) | Sustainable KGI (with QAS + Sparse Update) | Accuracy Difference |
|---|---|---|---|
| **Activity** | 0.9929 | 0.9763 | 0.02 |
| **Attribute** | 0.9573 | 0.7354 | 0.22 |
| **Direction** | 0.9405 | 0.4299 | 0.52 |
| **Entity Resolution** | 0.9594 | 0.5579 | 0.40 |
| **Location** | 0.9755 | 0.9436 | 0.03 |
| **Material** | 0.9766 | 0.9491 | 0.03 |
| **Reasoning** | 0.9929 | 0.9446 | 0.05 |
| **Sentiment** | 0.9987 | 0.9983 | 0 |
| **Size** | 0.9795 | 0.9548 | 0.02 |
| **State** | 0.9944 | 0.9899 | 0.02 |

Table 6 Accuracy score for fine-tuning KGI for sustainable KGI (with QAS + Sparse Update) and fine-tuned KGI

The KGI and sustainable KGI model is compared based on training time, energy consumption and CO2 Emission. The training time given in Table 7 is for fine-tuning the model for 10 epochs and the best model with highest accuracy is shown. Energy consumption is the total energy consumed for training the model and does not include the energy consumed for cooling. In order to find the carbon emission, the energy consumption* measured in kilowatt per hour(kWh) needs to be multiplied by carbon intensity measures in grams of CO2eq emitted per kWh. The carbon intensity for Ohio is 366 $gCO_2eq/kWh$ which is taken from electricity maps* resources. From Table 7, it can be observed that sustainable KGI is faster and has less carbon emission compared to fine-tuned KGI. Around 18% reduction across all the metrics is observed here. The values in bold are the lowest values. However, the overall accuracy for the sustainable KGI model is lower compared to fine-tuned KGI with values 85% and 98%, respectively. The accuracy of the sustainable KGI model is significantly lower for attribute, entity resolution and location gaps and performs well for other labels compared to fine-tuned KGI.

| **Fine-tuning Methods** | **Training time (Seconds)** | **Energy Consumption (kWh)** | **$CO_2$ Emission ($gCO_2eq/kWh$)** |
|---|---|---|---|
| **KGI (Without QAS and Sparse Update)** | 288.5 | 0.049 | 17.934 |
| **Sustainable KGI (with QAS + Sparse Update)** | **236.2** | **0.040** | **14.64** |
| **Reduction** | 52.3 (18 %) | 0.009 (18%) | 3.29 (18 %) |

Table 7 Comparing performance metrics for Sustainable KGI and Fine-tuned KGI

## 5.1 Knowledge Gap Identification Examples

**Example 1**

Question: Is the man to the left or to the right of the container made of glass?

Ground Truth KGs: ['Direction', 'Material']

Predicted KGs: ['Direction', 'Material']

Here, the predicted KGs match the ground truth KGs.

**Example 2**

Question: What is the person in front of the people doing?

Ground Truth KGs: ['Activity', 'Direction']

Predicted KGs: ['Activity', 'Direction', 'Entity Resolution']

Here, the predicted KGs do not match the ground truth. But the predicted labels seem more reasonable because the model has to identify the presence of ‘person’ and ‘people’ in the image which is associated with the ‘entity resolution’ gap.

* The energy consumption value is taken from XD Metrics on Demand (XDMoD) which is an open-source tool used by Ohio Supercomputer Center for analyzing job performance and utilization metrics. And https://www.electricitymaps.com/ is used for getting the carbon intensity factor. It is an open-source site which maps electricity to carbon emission for many parts of the world.

# Chapter 6. Future Work and Challenges

In this chapter, the summary of the work, challenges, and possible future directions are discussed.

From this work, *it can be concluded that the reasoning skill or cognitive capability also termed as knowledge gaps can be identified for the Visual Question Answering task*, to resolve the gaps. Further, *Tiny Machine Learning techniques are applied to build sustainable model, leading to 18% reduction in energy consumption and carbon di-oxide emission for model training*.

In this work, Visual Question Answering is used as a testbed to investigate Knowledge Gaps to build robust and flexible autonomous AI systems and addressed the following research question: *Can we build human cognitive capabilities into Artificially Intelligent systems with less computation and preserving accuracy?*

There were several challenges encountered while solving this problem statement. First, depending on each of the question types present in the dataset, knowledge gap tags differ for different visual question answering datasets, hence it is hard to train a single classifier model for all the datasets. To experiment this, a classifier model trained on GQA dataset was tested on TDIUC and vice versa, the performance metrics were low and zero for some classes. Secondly, from example 2 in section 5.1, it can be inferred that the ground truth annotations are not complete because the current approach uses rule-based mapping, and this type of reasonably predicted labels are observed all through the dataset particularly for 'Entity Resolution'. Finally, while performing multi-modal classification of question-

image features into knowledge gaps, the image features does not seem to contribute to the learning of the model, this is because of the underlying bias in the distribution which is observed in VQA datasets. For sustainable KGI model, applying quantization – aware scaling and sparse update for pre-training results in loss of information.

As a future work, this project can be extended by building a multi-modal classification model for CLEVR and TDIUC datasets. Implementing over-sampling or under-sampling for fine-tuning sustainable KGI model to prevent the performance drop. Quantization-Aware Scaling and Sparse Update methods can be implemented for tasks other than VQA for comparing knowledge gap identification with other tasks and performing detailed analysis. Additionally, to improve the efficiency of tiny neural networks, a tiny machine learning method called Network Augmentation (NetAug) [18] can be implemented to the Sustainable KGI model. NetAug is a novel training method that improves the performance of small neural networks by augmenting the network itself instead of the data, addressing under-fitting issues commonly found in tiny models.

# Bibliography


1. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision* (pp. 2425-2433)

2. Bajaj, G., Bandyopadhyay, B., Schmidt, D., Maneriker, P., Myers, C., & 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* (pp. 386-387).

3. Bajaj, G., Current, S., Schmidt, D., Bandyopadhyay, B., Myers, C. W., & Parthasarathy, S. (2022). Knowledge Gaps: A Challenge for Agent-Based Automatic Task Completion. *Topics in Cognitive Science*, *14*(4), 780-799.

4. Hudson, D. A., & Manning, C. D. (2019). Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6700-6709).

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

6. Kafle, K., & Kanan, C. (2017). An analysis of visual question answering algorithms. In *Proceedings of the IEEE international conference on computer vision* (pp. 1965-1973).

7. Lin, J., Zhu, L., Chen, W. M., Wang, W. C., Gan, C., & Han, S. (2022). On-device training under 256kb memory. *Advances in Neural Information Processing Systems*, *35*, 22941-22954.

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

9. Lin, J., Chen, W. M., Lin, Y., Gan, C., & Han, S. (2020). Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, *33*, 11711-11722.

10. Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

11. Cai, H., Gan, C., Zhu, L., & Han, S. (2020). Tinytl: Reduce activations, not trainable parameters for efficient on-device learning. *arXiv preprint arXiv:2007.11622*.

12. Haykin, S. (1998). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

13. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

14. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

15. O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

16. Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P. (2004). Integer quantization for deep learning inference: Principles and empirical evaluation. arXiv 2020. *arXiv preprint arXiv:2004.09602*.

17. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D. (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* (pp. 2704-2713).

18. Cai, H., Gan, C., Lin, J., & Han, S. (2021). Network augmentation for tiny deep learning. *arXiv preprint arXiv:2110.08890*.