Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.062837

ARTICLE

Performance vs. Complexity Comparative Analysis of Multimodal Bilinear Pooling Fusion Approaches for Deep Learning-Based Visual Arabic-Question Answering Systems

Sarah M. Kamel[1,*], Mai A. Fadel[2], Lamiaa Elrefaei[1,3] and Shimaa I. Hassan[1,4]

[1]Electrical Engineering Department, Faculty of Engineering at Shoubra, Benha University, Cairo, 11629, Egypt
[2]Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia
[3]Department of Computer and Systems Engineering, Faculty of Engineering and Technology, Badr University in Cairo (BUC), Cairo, 11829, Egypt
[4]Communication Systems Engineering Department, Faculty of Engineering, Benha National University, Obour, 11846, Qalyubia, Egypt

*Corresponding Author: Sarah M. Kamel. Email: sara.kamel@feng.bu.edu.eg

Received: 29 December 2024; Accepted: 04 March 2025; Published: 11 April 2025

ABSTRACT: Visual question answering (VQA) is a multimodal task, involving a deep understanding of the image scene and the question's meaning and capturing the relevant correlations between both modalities to infer the appropriate answer. In this paper, we propose a VQA system intended to answer yes/no questions about real-world images, in Arabic. To support a robust VQA system, we work in two directions: (1) Using deep neural networks to semantically represent the given image and question in a fine-grained manner, namely ResNet-152 and Gated Recurrent Units (GRU). (2) Studying the role of the utilized multimodal bilinear pooling fusion technique in the trade-off between the model complexity and the overall model performance. Some fusion techniques could significantly increase the model complexity, which seriously limits their applicability for VQA models. So far, there is no evidence of how efficient these multimodal bilinear pooling fusion techniques are for VQA systems dedicated to yes/no questions. Hence, a comparative analysis is conducted between eight bilinear pooling fusion techniques, in terms of their ability to reduce the model complexity and improve the model performance in this case of VQA systems. Experiments indicate that these multimodal bilinear pooling fusion techniques have improved the VQA model's performance, until reaching the best performance of 89.25%. Further, experiments have proven that the number of answers in the developed VQA system is a critical factor that affects the effectiveness of these multimodal bilinear pooling techniques in achieving their main objective of reducing the model complexity. The Multimodal Local Perception Bilinear Pooling (MLPB) technique has shown the best balance between the model complexity and its performance, for VQA systems designed to answer yes/no questions.

KEYWORDS: Arabic-VQA; deep learning-based VQA; deep multimodal information fusion; multimodal representation learning; VQA of yes/no questions; VQA model complexity; VQA model performance; performance-complexity trade-off

1 Introduction

Visual question answering (VQA) is about automatically answering a textual question based on the content of a given image or video, in a certain natural language. VQA is a multimodal task, that was recently

introduced in 2014 [1] and has gained significant interest over the last decade. Solving such a problem requires high-level perceptual capabilities for understanding the image and question semantics and cross-modal reasoning of language and vision. Therefore, it can be used as a key measurement for evaluating AI agents in both domains [2].

VQA systems generally involve four main modules, which are image features extraction, question features extraction, feature fusion, and answer prediction. Fig. 1 exhibits the general framework of VQA systems. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can improve the VQA model's performance, due to extracting robust and fine-grained representations for the input image and question, respectively. Several studies have adopted pre-trained VGGNet, GoogLeNet, AlexNet, InceptionNet, ResNet, and Faster R-CNN models for representing the input image, exploiting the principle of transfer learning [3]. Similarly, the Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) networks are widely used in the VQA field for question representation, as they can preserve long-term contextual information [3,4]. The feature fusion module is the core of VQA systems, where features from both modalities deeply interact to enable the VQA model to predict the best-matching answer correctly. Multimodal feature fusion is a challenging task that can greatly impact the overall model performance. This requires an efficient and expressive fusion technique that allows dense and high-level interactions to jointly embed features from the two different modalities and narrows the heterogeneity gap between their feature distributions. Hence, several studies compete to propose new multimodal fusion techniques or investigate different fusion techniques, aiming to achieve superior performance while maintaining minimal model complexity. Lastly, the answer prediction process can be formulated as a classification task over a pre-defined set of candidate answers, or as a sequence generation task for generating variable-length answers [3]. Most studies have adopted the multi-class classification approach over the top N most frequent answers in the training set.

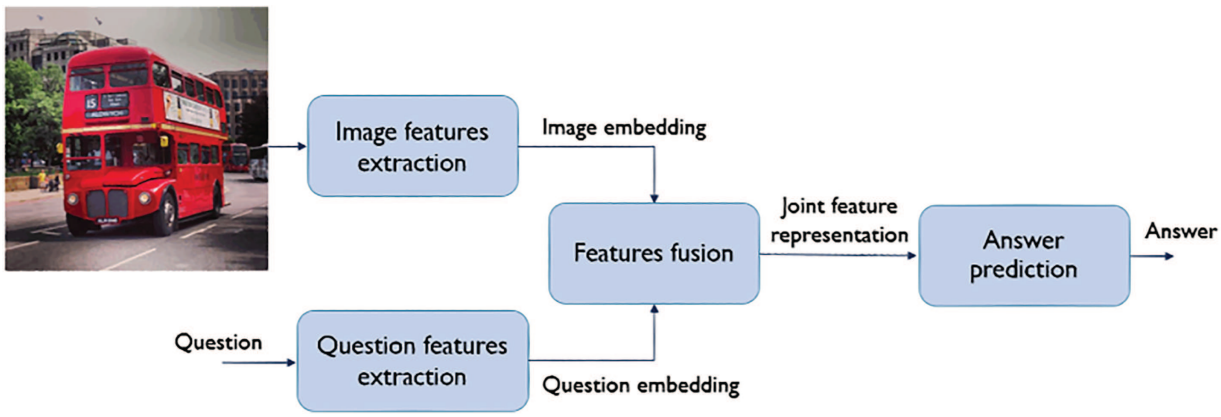

Figure 1: General framework of VQA systems

Feature fusion techniques play a vital role in the trade-off between the VQA model complexity and its performance [5]. This is because some fusion techniques tend to dramatically expand the joint feature space dimensionality, add multiple and large-sized fully connected layers during the fusion process, or cascade multiple fusion blocks several times. This is to achieve a highly discriminative and powerful joint feature representation. This not only would greatly improve the VQA accuracy, but it could also significantly increase the model size and the number of parameters to train. This seriously limits the applicability of such fusion

techniques for VQA models, due to requiring more powerful resources to train these VQA models with a tremendous memory consumption [6–9].

The fully parameterized bilinear pooling technique (FBP) [10] is a straightforward fusion technique that allows each element in one embedding to interact with every element in the other embedding, in a multiplicative manner. Although it allows rich interactions between the input embeddings, many studies have considered that the FBP technique is inapplicable for VQA models. This is because it leads to a massive number of learnable parameters, which makes the multimodal fusion module the most computationally expensive part of the VQA framework [5]. Recently, several bilinear pooling fusion techniques were proposed to solve the huge parameter space issue of the FBP technique, including the Multimodal Compact Bilinear Pooling (MCB) [11], Multimodal low-rank Bilinear pooling (MLB) [6], Multimodal Factorized Bilinear Pooling (MFB) [7], Multimodal Factorized High-order Pooling (MFH) [8], Multimodal Tucker Fusion (MUTAN) [12], Multimodal Local Perception Bilinear Pooling (MLPB) [9], and Bilinear Superdiagonal Fusion (BLOCK) [13].

These bilinear pooling fusion techniques have proven effective in reducing the number of model parameters, hence minimizing the model complexity, while preserving the model performance. However, in literature, all these bilinear pooling fusion techniques have been examined only for VQA systems designed to answer various question types with thousands of candidate answers (i.e., 3000 answers). But, what about VQA systems with a small set of candidate answers, as in the case of VQA systems designed to answer yes/no questions where there are only two candidate answers (i.e., two classes $l = 2$)? There is a severe lack of validating the efficiency of these multimodal bilinear pooling fusion techniques for VQA systems dedicated to yes/no questions, in terms of the model complexity and its performance as well. Hence, this article aims to answer several research questions about applying these bilinear pooling fusion techniques for this case of VQA systems, where VQA is formulated as a binary classification task over two candidate answers. These research questions are as follows:

1. Will the FBP technique remain inapplicable due to the number of its learnable parameters?
2. Will all these multimodal bilinear pooling fusion techniques accomplish their main objective of reducing the number of model parameters?
3. Will using these multimodal bilinear pooling fusion techniques improve the overall model performance?

In this work, we study the role of the utilized multimodal bilinear pooling fusion technique in the trade-off between the model complexity and the overall model performance, for VQA systems developed based on deep learning technology. We target VQA systems specialized in yes/no questions, to spotlight the impact of the number of answers in VQA systems on the effectiveness of these fusion techniques. Our contributions can be summarized as follows:

1. We propose a VQA system intended to answer yes/no questions about real-world images, in Arabic. Our Arabic-VQA system is developed on deep learning approaches, where the ResNet-152 and GRU models are employed for extracting discriminative and fine-grained representations for the given image and question, respectively. For feature fusion from both modalities, eight of the most popular multimodal bilinear pooling fusion techniques in the VQA field have been utilized, which are FBP, MCB, MLB, MFB, MFH, MUTAN, BLOCK, and MLPB. It is a novel research paper, since to the best of our knowledge, it is the first work to conduct a comprehensive study of all these fusion techniques in the case of VQA systems dedicated to yes/no questions. This is to validate their effectiveness for this case of VQA systems, in terms of their ability to improve the model performance and reduce the model complexity.

2. Proposing simple models for calculating the total number of learnable parameters for each multimodal bilinear pooling fusion technique.
3. Based on the model complexity and the achieved performance of the developed VQA models, several recommendations of these multimodal bilinear pooling fusion techniques are proposed for future VQA systems according to their number of answers.

The remaining of the paper is structured as follows: Since we study the role of multimodal fusion techniques used in VQA systems developed on deep learning technology, Section 2 provides a review of the most widely used fusion techniques in the VQA field with their different categories. Section 3 briefly covers the FBP technique, its huge parameter space issue, and how other bilinear pooling fusion techniques have tried to solve this issue. Section 4 demonstrates the framework and the deep learning approaches used to develop our Arabic-VQA system. It also presents simple models for calculating the number of model parameters for each bilinear pooling fusion technique. Section 5 exhibits the experimental results of all the developed Arabic-VQA models using all these bilinear pooling fusion techniques, showing which fusion techniques have contributed to improving the model's performance. It also discusses why some bilinear pooling fusion techniques have succeeded in achieving their main objective of reducing the model complexity while some other techniques have not, for VQA systems specialized in yes/no questions. Moreover, it presents recommendations for these multimodal bilinear pooling fusion techniques for future VQA systems according to their number of answers. Finally, Section 6 concludes the paper with a summary of the proposed Arabic-VQA system. It also summarizes our findings of which bilinear pooling fusion techniques provide good balances between model complexity and overall performance in the case of VQA systems intended to answer yes/no questions. This is in addition to providing an outlook for potential future work in the Arabic-VQA field.

2 Literature Review

Recently, several multimodal feature fusion techniques have been proposed for the VQA task. These techniques can be classified into three categories [4], according to the way of jointly embedding the extracted image and question representations into a common feature space. These fusion categories include simple vector operations performed on the image and question embeddings, non-linearly deep fusion procedures, or bilinear pooling fusion approaches. Fig. 2 presents the classification of multimodal fusion techniques that are widely utilized in the VQA research field.

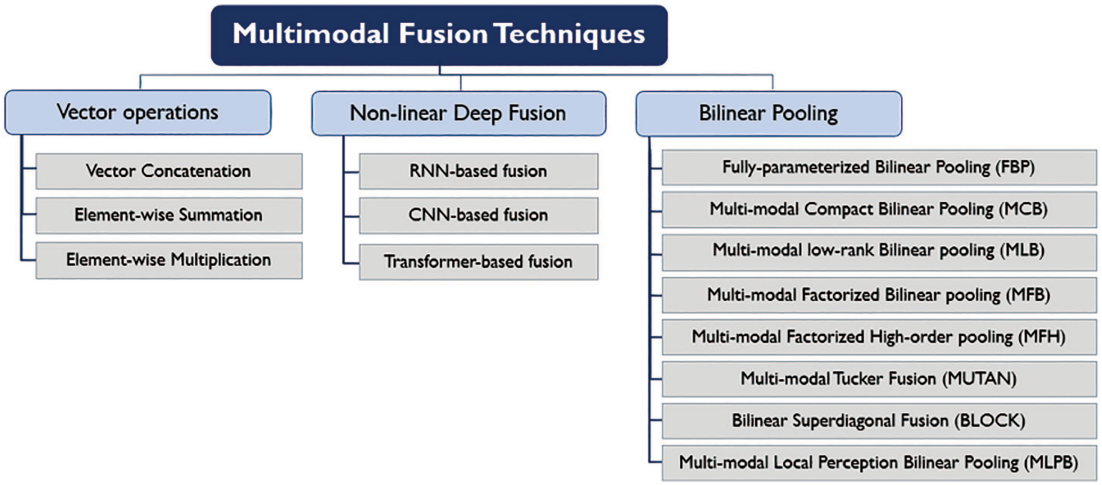

Figure 2: Classification of multimodal fusion techniques for VQA systems

Tables 1–3 summarize the related works, in terms of the utilized techniques during the VQA pipeline, and whether the answer prediction module is formulated as a classification task or a sequence generation task. Further, the size of the answers set for each VQA model is included (i.e., if mentioned in published articles) in these review tables. This is because this parameter matters in the classification-based VQA systems, where it represents the number of the model classes.

The simple vector operations category involves concatenating the two input embeddings or applying either element-wise summation or element-wise multiplication operations. This category has been a popular choice for feature fusion since the beginning of the VQA research field. These techniques can generate a joint feature representation, but they are not effective enough to fully capture the complex correlations between both modalities [11]. Element-wise summation and multiplication operations allow only elements in the same position in both embeddings to interact, which limits a rich interaction between features from both modalities. Further, these techniques directly integrate features from two different modalities, neglecting that their feature distributions may differ dramatically [7]. Thus, these simple vector operators have shown average performance among studies. Several deep learning-based VQA systems have utilized these simple fusion operators to integrate features from both modalities, whether they adopt an attention mechanism through their pipelines or not, as adopting the vector concatenation in [14–17], the element-wise summation in [18–20], and the element-wise multiplication in [21–25]. Similarly, many transformer-based VQA systems have employed this category of feature fusion techniques, whether they adopt a cross-modal attention mechanism through their pipelines or not, as utilizing the vector concatenation in [26–28], the element-wise summation in [29,30], and the element-wise multiplication in [31,32]. Table 1 summarizes the related work that used the simple vector operations category, where the three fusion techniques are abbreviated as vector concatenation (VC), element-wise summation (EWS), and element-wise multiplication (EWM).

Table 1: Summary of related work that used the simple vector operations category, in terms of the utilized techniques during the VQA pipeline, the answer prediction process, and the number of answers in each VQA model

VQA research ref.	Image representation	Question representation	Feature fusion	Answer prediction	Num. of candidate answers
Gao et al. (2015) [18]	GoogleNet	LSTM	EWS	Seq. Gen.	Not specified
Antol et al. (2015) [21]	VGGNet	LSTM	EWM	Classification	1000
Lu et al. (2016) [14]	ResNet	LSTM	VC	Classification	1000
Zhu et al. (2016) [22]	VGG-16	LSTM	EWM	Classification	5000
Xu et al. (2016) [19]	GoogleNet	Bag-of-Words	EWS	Classification	1000
Anderson et al. (2018) [23]	Faster R-CNN	GRU	EWM	Classification	3129
Ma et al. (2018) [15]	VGG-16 & ResNet-101	LSTM	VC	Classification	3000
Shi et al. (2018) [16]	Faster R-CNN & ResNet-152	LSTM	VC & MCB	Classification	Not specified
Gao et al. (2019) [24]	Faster R-CNN	GRU	EWM	Classification	Not specified
Gupta et al. (2021) [17]	Inception-Resnet-v2	Bi-LSTM	VC	Classification	Not specified

(Continued)

Table 1 (continued)

VQA research ref.	Image representation	Question representation	Feature fusion	Answer prediction	Num. of candidate answers
Yang et al. (2021) [29]	Faster-RCNN	BERT, XLNet, RoBERTa, ALBERT	EWS	Classification	Not specified
Alsaleh et al. (2022) [26]	ViT-16	Bert	VC	Seq. Gen.	Not specified
Yan et al. (2022) [31]	Faster-RCNN	Bert	EWM	Classification	3129
Kamel et al. (2023) [25]	VGG-16	LSTM	EWM	Classification	Two
Hackel et al. (2023) [32]	Deit_Tiny, MobileViT-S, XCiT Nano	BERT_TINY	EWM	Classification	1000
Bazi et al. (2023) [27]	ViT-32	Bert	VC	Seq. Gen.	Not specified
Liu et al. (2023) [28]	ViT-16	Bert	VC	Classification	99
Chen et al. (2024) [20]	Faster-RCNN	LSTM	EWS	Classification	Not specified
Huang et al. (2024) [30]	ResNet-152	TSE	EWS	Classification	1483

The FBP can generate richer joint representations than fusion techniques from the simple vector operations category, because of allows all pairwise interactions between the two input embeddings via applying the outer product operation. However, the major limitation of applying this technique for VQA systems is the high dimensionality of its feature space, which is usually in millions. For VQA systems having thousands of answers, this could lead to a massive model size with billions of model parameters that need to be trained, in the fusion stage only.

MCB [11] is one of the first attempts to compress the bilinear pooling operation. It enables a rich interaction between the image and question embeddings via approximating the outer product operation by computing their count sketches, followed by Fast Fourier Transformation (FFT). MCB can reduce the feature space dimensionality of the FBP from millions to thousands. However, this joint feature space is still required to be relatively high-dimensional, as the authors in [11] have set this hyperparameter to d_z = 16,000. This is to reach a reasonable performance and avoid biases [4]. Therefore, compared to the FBP, for a VQA model having thousands of answers, the number of model parameters will be decreased from billions to millions. MCB has been utilized in [11,33] for both attention and feature fusion purposes, while in [34] it has been used for feature fusion.

MLB [6] tries to factorize the 3-D huge weight matrix of the FBP technique into three low-rank matrices, to reduce the total number of the model parameters. MLB can achieve comparable performance to MCB while generating a much lower-dimensional joint feature representation than MCB. Hence, for a VQA model having thousands of answers, MLB can have much fewer parameters than MCB. However, MLB is slower to converge, and it is sensitive to the hyper-parameter d_z that represents its feature space dimensionality [4]. MLB has been utilized in [6,35] for both attention and feature fusion purposes, while in [36,37] a comparison between the MLB and MUTAN techniques has been conducted for the feature fusion purpose.

MFB [7] is inspired by the matrix factorization concept adopted in the MLB method, where the huge weight matrix of the fully parameterized bilinear pooling is decomposed into three weight matrices. However, unlike MLB, MFB tends to integrate the input embeddings in a higher dimensional space instead of a lower dimensional space, to capture richer interaction information. The higher the value of k for the MFB model, the more powerful the joint feature representation, and hence the higher the performance of the VQA model, but also the higher the number of model parameters. MLB can be viewed as a special case of MFB when $k = 1$. Therefore, MLB has much fewer parameters than MFB, but MFB can achieve a more powerful joint feature representation than MLB [4]. MFB has been utilized in [7,38–40] for both attention and feature fusion purposes.

MFH [8] is an extension of MFB, where multiple MFB blocks are cascaded. Although MFH multiplies the number of MFB parameters by p times, it can generate a more discriminative joint feature representation, hence increasing the VQA model robustness. This is because the cascading of multiple MFB blocks can capture higher-order interactions between the image and question features than just one MFB block [8]. MFH has been utilized in [8,41–43] for both attention and feature fusion purposes, while in [44] a comparison between the MFB and MFH techniques has been conducted for both purposes.

In [12], two versions of MUTAN techniques have been proposed. The first version of the MUTAN technique aims to utilize the outer product operation for integrating the image and question embeddings. This outer product operation causes a 3-D core matrix $\mathcal{T}_c$, that its dimensions d_v, d_q, and d_z directly impact the model complexity. To keep the number of the model parameters within an applicable range, the authors in [12] have restricted these dimensions to be relatively small (≤200). However, this could affect the model's robustness and cause a bottleneck in the modeling. Therefore, the authors of [12] have suggested a second version of the MUTAN technique, that tries to handle this bottleneck. The second version of the MUTAN technique presents an approximation of the 3-D core matrix $\mathcal{T}_c$, where the outer product operation is replaced by the element-wise multiplication operation. This enabled the authors to slightly increase the values of the three dimensions to $d_v = d_q = d_z = 360$, thus still maintaining the model complexity under control. MUTAN has been utilized in [45] as a feature fusion technique, while it has been utilized in [12,46,47] for both attention and feature fusion purposes. In [48], a comparison between the MLB, MUTAN, and BLOCK techniques has been conducted for the feature fusion purpose, where MUTAN has shown the best performance.

BLOCK [13] is an extension of MUTAN, where some modifications have been introduced to both versions of the MUTAN technique. Like the first version of MUTAN, the first version of BLOCK also applies the outer product operation for feature fusion. This version of the BLOCK technique tries to solve the bottleneck of the first version of the MUTAN technique. This is achieved by dividing the image and question representations $\hat{v}$ and $\hat{q}$ into small chunks and hence dividing the 3-D core matrix $\mathcal{T}_c$ of MUTAN into several smaller core blocks D_c. This enables the $\hat{v}$ and $\hat{q}$ representations to be of relatively high dimensions, thus improving the model robustness, while minimizing the model complexity as much as possible. In the second version of BLOCK, the outer product operation performed between the image and question chunks is replaced by the element-wise multiplication operation. This replacement helps in reducing the number of model parameters for VQA models having thousands of answers. BLOCK has been utilized in [13] for both attention and feature fusion purposes, while in [49–51] it has been used for feature fusion.

MLPB [9] is another attempt to utilize the outer product operation for integrating the image and question embeddings without converting it into the element-wise multiplication operation as done in the other bilinear pooling fusion techniques. To achieve this while reducing the number of the model parameters, the outer product operation is performed between a low-dimensional image/question kernel with small question/image clips separately, then concatenating all results. This results in multiple 3-D weight matrices, which could significantly increase the number of model parameters. To solve this issue, the authors in [9] have suggested sharing the same 3-D weight matrix for all the performed outer product operations. However,

the image kernel v_k and the question kernel q_k are restricted to be relatively low-dimensional (≤150), to maintain the number of the model parameters within an applicable range. Table 2 summarizes the related work that used the bilinear pooling fusion category, where it can be noticed that all these techniques are investigated only for VQA models having thousands of answers.

Table 2: Summary of related work that used the bilinear pooling fusion category, in terms of the techniques utilized during the VQA pipeline, the answer prediction process, and the number of answers in each VQA model

VQA research ref.	Image representation	Question representation	Feature fusion	Answer prediction	Num. of candidate answers
Fukui et al. (2016) [11]	ResNet-152	LSTM	MCB	Classification	3000
Kim et al. (2017) [6]	ResNet-152	GRU	MLB	Classification	2000
Yu et al. (2017) [7]	ResNet-152	LSTM	MFB	Classification	3000
Ben-younes et al. (2017) [12]	ResNet-152	GRU	MUTAN	Classification	2000
Yu et al. (2018) [8]	ResNet-152	LSTM	MFH	Classification	3000
Lao et al. (2018) [9]	ResNet	LSTM	MLPB	Classification	3000
Shi et al. (2018) [16]	Faster R-CNN & ResNet-152	LSTM	VC & MCB	Classification	Not specified
Abacha et al. (2018) [33]	ResNet-152 & ResNet-50	LSTM	MCB	Classification	Not specified
Su et al. (2018) [35]	ResNet-152	LSTM	MLB	Classification	2000
Li et al. (2018) [46]	ResNet-152	LSTM	MUTAN	Classification	2000
Ben-younes et al. (2019) [13]	Faster R-CNN	GRU	BLOCK	Classification	3000
Osman et al. (2019) [34]	Faster R-CNN	LSTM	MCB	Classification	3000
Vu et al. (2019) [37]	ResNet-152	BERT	MLB & MUTAN	Classification	1700
Liu et al. (2019) [44]	ResNet-152	LSTM	MFB & MFH	Classification	Not specified
Jung et al. (2020) [41]	VGG-16	BioBERT	MFH	Classification	Not specified
Zheng et al. (2020) [49]	VGG-16	BERT	BLOCK	Classification	Not specified
Sharma et al. (2021) [38]	ResNet-152	BERT	MFB	Classification & Seq. Gen.	Not specified
Li et al. (2021) [39]	ResNet-34	LSTM	MFB	Classification	Not specified
Li et al. (2021) [42]	VGG-16	BioBERT	MFH	Classification	Not specified
Bozinis et al. (2021) [47]	ResNet-50	GRU	MUTAN	Classification	Not specified
Li et al. (2022) [40]	ResNet-34	LSTM	MFB	Classification	3794
Shuang et al. (2022) [45]	Faster R-CNN	GRU	MUTAN	Classification	Not specified
Miao et al. (2022) [50]	Faster R-CNN	GRU	BLOCK	Classification	Not specified
Lu et al. (2023) [36]	ResNet-152	GRU	MLB & MUTAN	Classification	Not specified
Upadhyay et al. (2023) [43]	VGG-19	BioBERT	MFH	Classification	Not specified
Mohamud et al. (2023) [51]	Faster R-CNN	GRU	BLOCK	Classification	3000

(Continued)

Table 2 (continued)

VQA research ref.	Image representation	Question representation	Feature fusion	Answer prediction	Num. of candidate answers
Mao (2024) [48]	ResNet-152	LSTM	MLB, MUTAN, BLOCK	Classification	Not specified

Other than linear and bilinear pooling fusion techniques, another fusion mechanism is about non-linear fusing the image and question features using deep neural networks. In the RNN-based fusion strategy, as in [52,53], the image embedding extracted from a CNN is projected to the question word embedding space and treated as one of the question words. Then, the question word embeddings and the image embedding are fed together to the utilized RNN to handle the semantic features of the input question and generate a joint feature representation, simultaneously. Despite the simplicity of the VQA model design, this strategy is not efficient enough to capture the complex correlations between both modalities. This is because the image effect will vanish at each step of the utilized RNN [54]. To tackle this issue, the CNN-based fusion strategy was proposed in [54]. An end-to-end CNN-based VQA model was developed, where the image and question embeddings are both extracted by CNNs and then fused by a multimodal convolution layer. However, CNNs cannot process the question's sequential information well. Hence, only a few early studies have utilized RNN and CNN-based fusion strategies.

Recently, some transformer-based VQA studies have adopted multimodal transformers for encoding and fusing the input image and question simultaneously, as in [55–59]. In this strategy, the mono-modal representations are kept extremely simple, while the complex processing is performed in a BERT-like transformer encoder. This fusion strategy is like the RNN-based fusion strategy to some extent, where the image embedding extracted from a CNN is projected to the question word embedding space, and both embeddings are fed together to the utilized multimodal transformer encoder. However, recent studies tend to utilize a specialized transformer for each modality separately, to obtain rich mono-modal representations, and then use one of the fusion approaches from the other fusion categories, as in the transformer-based VQA systems mentioned earlier in Tables 1 and 2. Table 3 summarizes the related work that used the non-linear deep fusion category.

Table 3: Summary of related work that used the non-linear deep fusion category, in terms of the utilized techniques during the VQA pipeline, the answer prediction process, and the number of answers in each VQA model

VQA research ref.	Image representation	Question representation	Feature fusion	Answer prediction	Num. of candidate answers
Malinowski et al. (2015) [52]	GoogleNet	FC embedding layer	LSTM-based fusion	Seq. Gen.	Not specified
Ren et al. (2015) [53]	VGG-19	Skip-gram word embedding	LSTM-based fusion	Classification	Not specified
Ma et al. (2015) [54]	VGGNet	Sentence CNN	CNN-based fusion	Classification	Not specified

(Continued)

Table 3 (continued)

VQA research ref.	Image representation	Question representation	Feature fusion	Answer prediction	Num. of candidate answers
Khare et al. (2021) [55]	ResNet-152	BERT wordpiece tokenizer	MMBERT	Classification	Not specified
Silva et al. (2022) [56]	EfficientNetV2	BERT wordpiece tokenizer	Real-Former Trans-former	Classification	Not specified
Seenivasan et al. (2022) [57]	ResNet-18	BERT wordpiece tokenizer	VisualBERT ResMLP	Classification & Seq. Gen.	Not specified
Siebert et al. (2022) [58]	ResNet-152	BERT wordpiece tokenizer	VisualBERT	Classification	Not specified
Naseem et al. (2023) [59]	ResNet50	Bi-LSTM	Transformer encoder	Seq. Gen. using trans-former decoder	Not specified

From Tables 1–3, we can observe that most VQA studies have employed deep learning approaches to develop high-performance VQA models. However, only a few studies have investigated different multimodal fusion techniques for capturing the relevant correlations from both modalities, with a maximum investigation of three bilinear pooling fusion techniques as in [48]. Moreover, these multimodal bilinear pooling fusion techniques have been widely investigated only for VQA models containing thousands of answers. So far, there is no evidence of how efficient these fusion techniques are for VQA systems specialized in answering yes/no questions, in terms of the model complexity and its performance as well. Therefore, in this work, we extensively analyze the effectiveness of eight bilinear pooling fusion techniques for this case of VQA models, which are FBP, MCB, MLB, MFB, MFH, MUTAN, BLOCK, and MLPB. For this purpose, we propose an Arabic-VQA system specialized in answering yes/no questions, that is developed using deep learning approaches.

3 Preliminaries of Multimodal Bilinear Pooling Fusion

This section presents a brief description of the FBP technique and its huge parameter space issue. It also discusses how several bilinear pooling fusion techniques have attempted to reduce the dimensionality of the generated bilinear vector z, hence solving this parameter space issue.

3.1 FBP

Given an image embedding $v \in R^m$ and a question embedding $q \in R^n$, the fully parameterized bilinear pooling (FBP) [10] is simply about applying the outer product operation between the two vectors. This technique tends to quadratic expand the feature space dimensionality, producing a bilinear vector $z \in R^{m \times n}$.

This bilinear vector z is then passed through a fully connected layer to obtain the final joint representation $y \in R^l$. The FBP operation is formulated as follows [11]:

$$y = W\left[v \otimes q\right] = v^T W q \tag{1}$$

where $W \in R^{m \times n \times l}$ is the weight matrix, l typically equals the number of classes in the VQA model and $\otimes$ denotes outer product operation. For example, suppose a VQA model that extracts an image embedding of size $m = 2048$ and a question embedding of size $n = 2048$, that has a set of candidate answers of size $l = 3000$. The number of learnable parameters for the fusion stage will exceed 12.5 billion parameters, which is a huge space of learnable parameters. Hence, several studies have considered that the direct application of fully parameterized bilinear pooling in VQA is inapplicable because such a huge number of parameters leads to tremendous resource consumption and a very expensive computational cost. Numerous attempts were proposed to reduce the feature space dimensionality of the bilinear pooling, hence reducing the huge parameter space issue. These methods are briefly described in the following sub-sections.

3.2 MCB

MCB [11] presents an approximation of the outer product operation, by exploiting two different properties. The first property is that the count sketch of the outer product of two vectors is equal to the convolution of their count sketches [11]. Thus, the outer product between the image and question embeddings is approximated as follows [11]:

$$\mathbb{C}\left(v \otimes q\right) = \mathbb{C}(v) \ * \ \mathbb{C}(q) \tag{2}$$

where $\mathbb{C}$ denotes the count sketch projection function and $*$ implies the convolution operator. The second property is that the convolution in the time domain is equal to element-wise multiplication (i.e., Hadamard product) in the frequency domain [11]. The output of frequency domain multiplication is then transformed back to the original domain by applying inverse FFT, to obtain the bilinear vector $z \in R^{d_z}$. Therefore, Eq. (2) is re-formulated as follows [11]:

$$z = \mathbb{C}\left(v \otimes q\right) = FFT^{-1}(FFT\left(\mathbb{C}(v)\right) \odot FFT\left(\mathbb{C}(q)\right)) \tag{3}$$

where $\odot$ implies element-wise multiplication, $\mathbb{C}(v) \in R^{d_z}$ and $\mathbb{C}(q) \in R^{d_z}$ are the count sketches of the image and question embeddings, and d_z is a hyper-parameter that refers to the feature space dimensionality. This bilinear vector z is then normalized into z_{norm} using the signed square root normalization and the l_2 normalization operations. Lastly, the bilinear vector z_{norm} is passed through a fully connected layer to generate the final joint representation $y \in R^l$.

3.3 MLB

MLB [6] decomposes the 3-D weight matrix of bilinear pooling into the multiplication of three smaller 2-D weight matrices. Firstly, the image embedding $v \in R^m$ and the question embedding $q \in R^n$ are linearly projected to a lower dimensional space, using two projection matrices $W_v \in R^{m \times d_z}$ and $W_q \in R^{n \times d_z}$, respectively. d_z is a hyper-parameter that represents the feature space dimensionality and should be $d_z \leq \min(m, n)$. Then, the projected vectors $\hat{v} \in R^{d_z}$ and $\hat{q} \in R^{d_z}$ are integrated by using the Hadamard product. Lastly, the resultant bilinear vector $z \in R^{d_z}$ is linearly projected by a third weight matrix $W_z \in R^{d_z \times l}$, to obtain the final joint representation $y \in R^l$. The MLB operation can also be represented as follows [6]:

$$y = W_z^T \left(W_v^T v \odot W_q^T q\right) \tag{4}$$

3.4 MFB

MFB [7] tries to solve the convergence issue of the MLB, by adding a sum-pooling operation on the integrated feature vector. The MFB process is divided into two stages:

1. Expansion stage, where the image and question embeddings are expanded to a higher dimensional space, using two projection matrices $W_v \in R^{m \times ko}$ and $W_q \in R^{n \times ko}$, respectively. Then, the projected vectors are integrated by using the Hadamard product. To prevent overfitting, a dropout layer is added after integrating features from both modalities.
2. Squeezing stage, where the high dimensional multiplication result $x \in R^{ko}$ is squeezed by sum pooling into a bilinear vector $z \in R^o$. This is to compress the feature space dimensionality again. The bilinear vector z is then normalized into z_{norm} using the signed square root normalization and the l_2 normalization operations.

Finally, the bilinear vector z_{norm} is linearly projected by a third weight matrix $W_z \in R^{o \times l}$, to obtain the final joint representation $y \in R^l$. The MFB process is formulated as follows [7]:

$$z = SumPooling(W_v^T v \odot W_q^T q, k) \tag{5}$$

where the function $SumPooling(x, k)$ stands for performing sum pooling over x by a 1-D non-overlapped window of size k. k and o are hyper-parameters, where k is the factorization rank of the two projection matrices and o represents the feature space dimensionality.

3.5 MFH

MFH [8] is the cascading of multiple MFB blocks, where the same procedures of MFB fusion are performed p number of times. For the ith MFB block, the Hadamard product is performed between the expanded inputs of the current block and the expansion output of the $(i-1)$th MFB block. The bilinear vector $z \in R^{po}$ is obtained by concatenating all the normalized bilinear vectors z_{norm} generated from all the p MFB blocks. Lastly, this bilinear vector is linearly projected by a weight matrix $W_z \in R^{p \times o \times l}$, to obtain the final joint representation $y \in R^l$. Hence, MFB is a special case of MFH_p, when $p = 1$.

3.6 MUTAN

In [12], the authors have proposed two versions of the MUTAN technique. In the first version, there are no structural constraints enforced on the 3-D core matrix $\mathcal{T}_c$. In the second version, a fixed rank constraint is enforced on the core matrix. The two versions are briefly described in the following two sub-sections.

3.6.1 MUTAN without a Fixed Rank Constraint

MUTAN [12] decomposes the huge 3-D weight matrix of bilinear pooling into the mode product of three 2-D projection matrices and a small 3-D core matrix, using the Tucker decomposition. Firstly, the image embedding $v \in R^m$ and the question embedding $q \in R^n$ are linearly projected to a lower dimensional space, using two projection matrices $W_v \in R^{m \times d_v}$ and $W_q \in R^{n \times d_q}$, respectively. To integrate the two projected embeddings $\hat{v} \in R^{d_v}$ and $\hat{q} \in R^{d_q}$ into a bilinear vector $z \in R^{d_z}$ using the outer product operation, a 3-D weight matrix $\mathcal{T}_c \in R^{d_v \times d_q \times d_z}$ is needed. This core matrix $\mathcal{T}_c$ is responsible for capturing the bilinear interactions between the projected embeddings $\hat{v}$ and $\hat{q}$. Lastly, the resultant bilinear vector z is linearly projected by a weight matrix $W_z \in R^{d_z \times l}$, to obtain the final joint representation $y \in R^l$. This version of MUTAN technique is represented as follows [12]:

$$y = ((\mathcal{T}_c \times_1 (q^T W_q)) \times_2 (v^T W_v)) \times_3 W_z \tag{6}$$

where the operator $\times_i$ denotes the ith mode product between matrices. d_v, d_q, and d_z are hyper-parameters that represent the dimensions of the core matrix.

3.6.2 MUTAN with Enforcing a Fixed Rank Constraint

In this MUTAN version, an additional constraint is imposed on the core matrix $\mathcal{T}_c$, where each slice of the core matrix is forced to have a constant rank R. By introducing this structural constraint on $\mathcal{T}_c$, each slice $\mathcal{T}_c[:,:,k]$ where $\forall k \in [1, d_z]$ is re-defined as the sum of the outer product of R weight vectors $a_r^k \in R^{d_q}$ and $b_r^k \in R^{d_v}$, which is represented as follows [12]:

$$\mathcal{T}_c[:,:,k] = \sum_{r=1}^{R} a_r^k \otimes b_r^k \tag{7}$$

For each value r where $\forall r \in [1, R]$, two projection matrices $A_r \in R^{d_q \times d_z}$ and $B_r \in R^{d_v \times d_z}$ can be defined, where $A_r[:,k] = a_r^k$ and $B_r[:,k] = b_r^k$. Thus, it can be summarized that the 3-D core matrix $\mathcal{T}_c$ is replaced by two groups of matrices, namely A and B. Each group consists of R projection matrices, such as $A = [A_1, \ldots, A_r, \ldots, A_R]$ and $B = [B_1, \ldots, B_r, \ldots, B_R]$. The low-dimensional question and image representations $\hat{q}$ and $\hat{v}$ are projected R times, each time with two different projection matrices A_r and B_r, respectively, and then integrated by the Hadamard product. Hence, the bilinear vector $Z \in R^{d_z}$ is formulated as follows [12]:

$$Z = \sum_{r=1}^{R} Z_r \tag{8}$$

$$Z_r = (\hat{q}^T A_r) \odot (\hat{v}^T B_r) \tag{9}$$

3.7 BLOCK

BLOCK [13] technique can be considered as a modified edition of the MUTAN technique. Just like MUTAN, two versions of the BLOCK technique were proposed. In the first version, there are no constraints enforced on the 3-D core blocks D_c. In the second version, a fixed rank constraint is enforced on the core blocks. For both cases of enforcing or not enforcing a fixed rank constraint on the core matrices, MUTAN can be viewed as a special case of BLOCK when $C = 1$, while using the same values of the other hyper-parameters. The two versions are briefly described in the following two sub-sections.

3.7.1 BLOCK without a Fixed Rank Constraint

BLOCK decomposes the 3-D core matrix $\mathcal{T}_c$ of the MUTAN technique into C core blocks. Each core block is responsible for capturing the bilinear interactions between small chunks of the image and question embeddings. Firstly, the image embedding $v \in R^m$ and the question embedding $q \in R^n$ are linearly projected to a lower dimensional space, using two projection matrices $W_v \in R^{m \times ck_v}$ and $W_q \in R^{n \times ck_q}$, respectively. The projected image and question feature vectors $\hat{v} \in R^{Ck_v}$ and $\hat{q} \in R^{Ck_q}$ are divided into C chunks of sizes k_v and k_q, respectively. Each pair of corresponding chunks of the image and question features are integrated via the outer product operation, and then the product result $x_c \in R^{k_v \times k_q}$ is linearly mapped to $z_c \in R^{k_z}$ using a 3-D core block $\mathrm{D}_c \in R^{k_v \times k_q \times k_z}$. The bilinear vector $z \in R^{Ck_z}$ is obtained by concatenating all the z_c vectors. Lastly, this bilinear vector is linearly projected by a weight matrix $W_z \in R^{Ck_z \times l}$, to generate the final joint representation $y \in R^l$. For each chunk c where $\forall c \in [1, C]$, the BLOCK operation can be written as follows [13]:

$$z_c = \mathrm{D}_c \times_1 \hat{q}_{ck_q:(c+1)k_q} \times_2 \hat{v}_{ck_v:(c+1)k_v} \tag{10}$$

$$z = [z_1, \ldots, z_c, \ldots, z_C] \tag{11}$$

3.7.2 BLOCK with Enforcing a Fixed Rank Constraint

To reduce the number of the model parameters while preserving the model robustness, another version was proposed where a constraint is imposed on the rank of the slice matrices of each core block D_c. Just like the MUTAN technique, the outer product operation between each pair of corresponding image and question chunks is replaced by element-wise multiplication, and each core block D_c is replaced by two groups of matrices, namely A_c and B_c. For each chunk c, each group consists of R projection matrices, such as $A_c = [A_1^c, \ldots, A_r^c, \ldots, A_R^c]$ and $B_c = [B_1^c, \ldots, B_r^c, \ldots, B_R^c]$.

3.8 MLPB

MLPB [9] technique is composed of two phases, namely the question-kernel-based pooling phase and the image-kernel-based pooling phase. These two phases of operation are symmetric. In the question-kernel-based pooling phase, the question embedding $q \in R^n$ is linearly projected to a much lower dimensional space, using a weight matrix $W_q \in R^{n \times p}$. This is to obtain the question kernel $q_k \in R^p$, where p is a hyper-parameter that represents the question-kernel dimensionality. This question kernel slides over the image embedding with a pre-defined stride s, and interacts with the corresponding image clip via the outer product operation. Thereafter, for each sliding step t, the product result $x_q^t \in R^{p \times p}$ is linearly projected to a lower dimensional space using a weight matrix $W_{x_q}^t \in R^{p \times p \times o}$, to obtain the joint feature vector $h_q^t \in R^o$ of the current step t. Lastly, the question-kernel-based fusion representation z_q is the concatenation of the joint feature vectors from all these steps, which is represented as follows [9]:

$$z_q = [h_q^1, \ldots, h_q^t, \ldots, h_q^T] \in R^{o \times T_q} \tag{12}$$

where T_q is the total number of sliding steps needed for passing the question kernel over the entire image embedding with the stride s, and o is a hyper-parameter that represents the output feature space dimensionality for each sliding step t. The same procedures are performed in the image-kernel-based pooling phase, where the image kernel slides over the question embedding to interact with the small question clips. The bilinear vector z is obtained by concatenating the two output representations from both phases together, as represented in Eq. (13) [9]. Finally, this bilinear vector is passed through a fully connected layer to obtain the final joint representation $y \in R^l$.

$$z = [z_q, z_v] \in R^{o \times T_q + o \times T_v} \tag{13}$$

4 Proposed Methodology

The proposed Arabic-VQA system consists of five main modules, all of which are developed using deep learning approaches. These modules are: (1) image features extraction, (2) question pre-processing, (3) question features extraction, (4) feature fusion, and (5) answer prediction. Fig. 3 presents the framework of our Arabic-VQA system. The proposed Arabic-VQA system is developed using the VAQA dataset [25]. For non-Arabic native readers, the English translation of the input question and the output answer are provided in the lower left corner of the same figure.

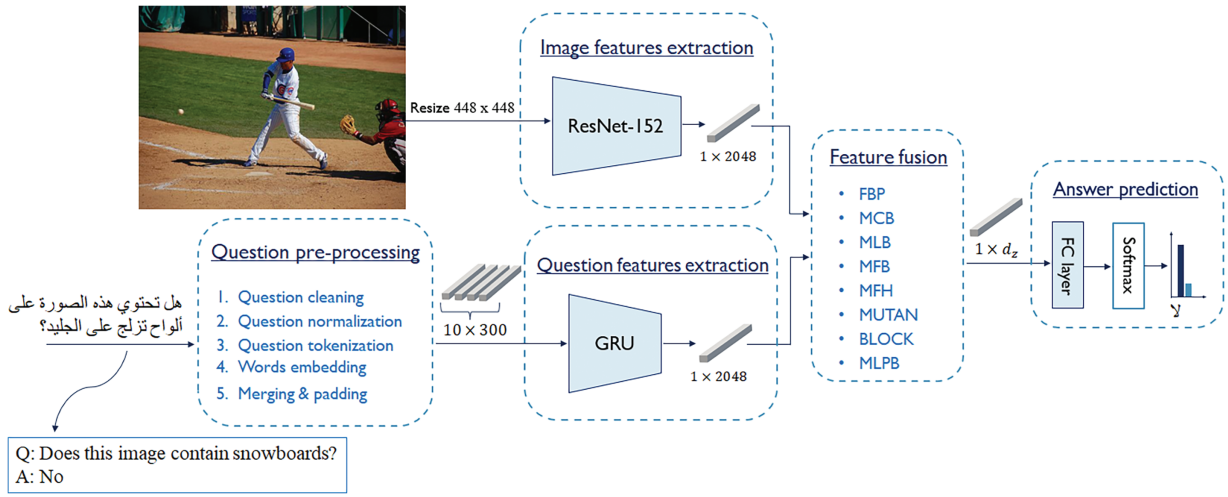

Figure 3: Framework of the proposed Arabic-VQA system

4.1 Image Features Extraction

In this module, we have adopted the Image Level (IL) representation approach, where the image embedding is extracted from the whole image. According to [60], the VGG and ResNet are the top-used networks among the VQA studies for image representation. Although ResNet-152 [61] is much deeper than both VGG-16 and VGG-19, it has a much lower model complexity [61]. This is due to the use of global average pooling instead of the fully connected layers in VGG networks. VGG networks not only have more parameters compared to ResNet-152, and therefore take longer to train, but they also have lower performance [61]. Hence, a ResNet-152 model that is pre-trained on the ImageNet dataset [62] is adopted, where its parameters remain frozen during train our Arabic-VQA models. The last fully connected layer of classification in the ResNet-152 model is discarded and the image embedding is obtained as the output of the last pooling layer, as shown in Fig. 4. The given image is initially re-scaled to 448 × 448 pixels and then passed through the ResNet-152 network to obtain the image embedding as a 2048-dim feature vector. This feature vector is then normalized by applying l_2 normalization operation. This is to restrict the values to being within a narrow range.

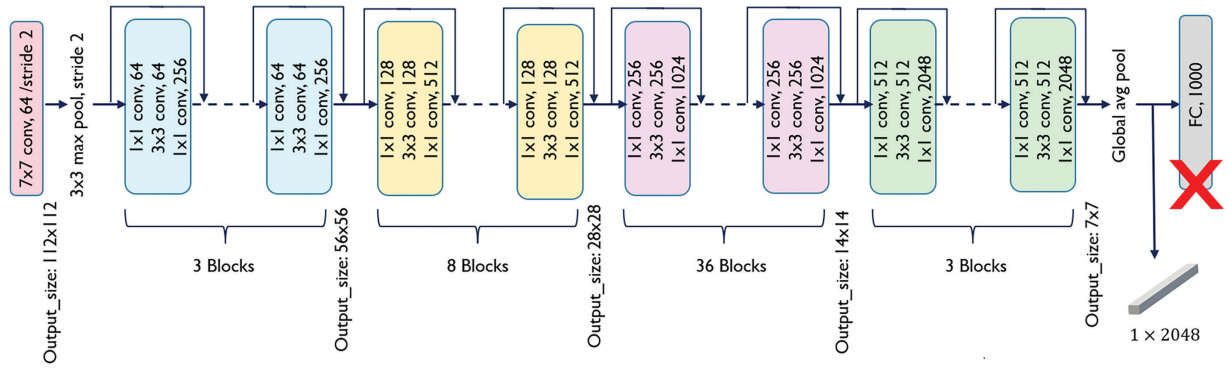

Figure 4: Block diagram of the ResNet-152 model

4.2 Question Channel

Arabic questions from the VAQA dataset have three different tasks: COCO object existence verification, COCO super-category existence verification, and image scene recognition [25]. All questions in the VAQA dataset were automatically generated using three grammatically structured formulas, which were represented as follows [25]:

$$object\_name + [noun + modal\_verb] \mid demonstrative\_pronoun \mid main\_verb + Question\_tool$$

$$location + \text{الصورة} + demonstrative\_pronoun + noun + \text{تم} + Question\_tool$$

$$adjective + \text{مشهد} + demonstrative\_pronoun + Question\_tool$$

In Arabic, there are only two question tools for yes/no questions, namely "هل" and "أ". On the other hand, several values were used for each of the other question components to generate questions of great diversity, as discussed in [25]. This resulted in 110 unique question templates being used through the VAQA dataset for yes/no question generation [25].

Through the question channel, the semantic meaning of the input question should be well understood to support the VQA system to infer the appropriate answer. This is done during both modules of question pre-processing and question feature representation, starting from cleaning and normalizing the raw question, passing through understanding the meanings of each word, followed by capturing the sequential relationships between the question words. The following two sub-sections describe the operations of the two question modules in detail.

4.2.1 Question Pre-Processing

In this module, the raw Arabic question is pre-processed through five steps, which are briefly described as follows:

1. Question cleaning, where all non-alphabetic symbols in the given question are eliminated, such as diacritics and question marks.
2. Question normalization, where letters that can come in various forms are unified into a single form.
3. Question Tokenization, where the question is segmented into individual tokens. The special case of separating the question tool "أ" into a distinct token is considered during Arabic yes/no questions tokenization, as proven in [25]. This is because the question tool "أ" is equivalent to the question tool "هل" which always comes as a separate word but "أ" does not.
4. Word embedding, where each question word is encoded into a numerical representation that expresses its meaning and context. For this purpose, the SG model from the pre-trained AraVec2.0 tool [63] is utilized after fine-tuning it with all questions of the VAQA dataset to handle the missing question words. This is to represent each question word as a 300-dim word embedding vector.
5. Merging and padding, where long questions are trimmed to a predefined fixed question length of $F = 10$ words, while shorter questions are padded. This is followed by merging all the question's word embeddings into a single matrix.

4.2.2 Question Features Extraction

Usually, one of the recurrent neural networks (RNNs) is used to capture the semantic features and sequential relationships between the question words. LSTMs and GRUs are widely used for question features extracted in the VQA field. GRU has fewer gates and fewer parameters compared to LSTM, as represented in Fig. 5. This makes it faster to train and execute while using less memory and retaining the same ability to

preserve long-term contextual information as LSTM. Hence, a one-layer unidirectional GRU (1-layer Uni-GRU) [64] with an internal hidden state of dimension 2048 is adopted for Arabic-question representation, where the question embedding is obtained as the output of the last hidden state in this hidden layer. Thereafter, a dropout process with a ratio of $p = 0.3$ is applied to prevent overfitting.

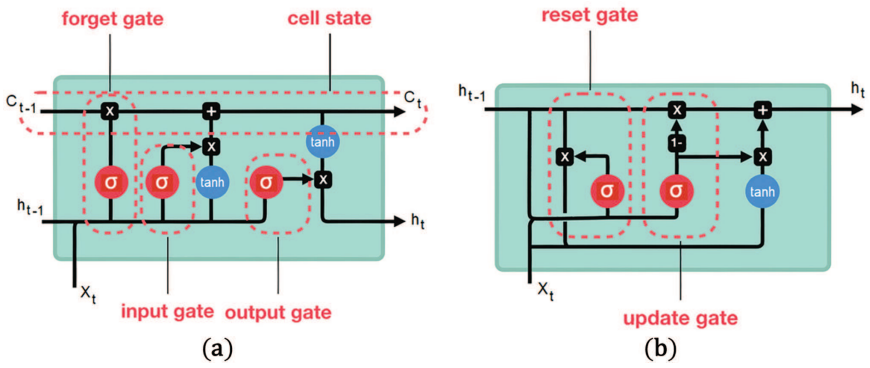

Figure 5: Difference between LSTM and GRU, (a) LSTM architecture and (b) GRU architecture

4.3 Feature Fusion

In this module, eight multimodal bilinear pooling fusion techniques are utilized, which are the FBP, MCB, MLB, MFB, MFH, MUTAN, BLOCK, and MLPB. This is to explore how efficiently these fusion techniques will perform in the case of VQA systems intended to answer yes/no questions and determine the most effective technique for this case of VQA systems. For conducting an impartial comparison between all the utilized bilinear pooling fusion techniques, we have followed the hyperparameter values that have been used for each technique in its published article. For the same reason, both normalization and dropout operations are generalized for all the utilized bilinear pooling fusion techniques. Therefore, the bilinear vector z is normalized into z_{norm} using the signed square root normalization and l_2 normalization operations. This is to restrict the values of the bilinear vector generated from each fusion technique to be within a narrow range, resulting in more stable VQA models [7]. Thereafter, the dropout process with a ratio of $p = 0.1$ is applied to prevent overfitting. On the other hand, the dropout and normalization operations are already included as part of the framework of the MFB and MFH fusion techniques.

4.3.1 FBP

The FBP is included in our experiments for two reasons: (1) to validate its applicability for the VQA systems containing a small set of candidate answers, and (2) to be a reference for the other fusion techniques, to assess their abilities to improve the model performance and reduce the model complexity. Fig. 6 graphically represents the fully parameterized bilinear pooling operation, including all generated feature vectors step by step with their dimension values, and all weight matrices needed during fusion. The fully parameterized bilinear pooling doesn't require any hyper-parameters during the fusion process. Including the bias inputs, the number of learnable parameters for the fully parameterized bilinear pooling is calculated as follows:

$$\text{Num. of } FBP\ parameters = size\ of\ W = (m \times n + 1) \times l \tag{14}$$

Hence, the number of parameters for the fusion module in our Arabic-VQA system using the fully parameterized FBP is 8,388,610 parameters.

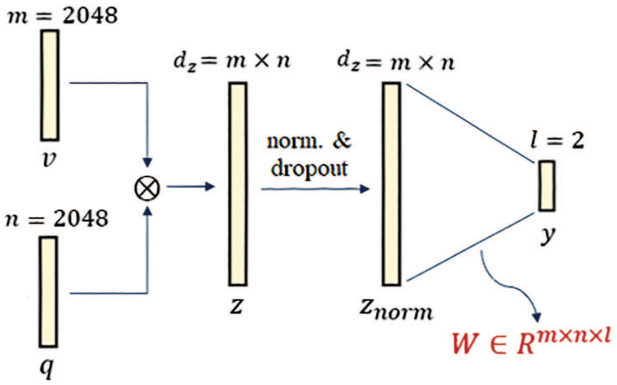

Figure 6: Graphical representation of the fully parameterized bilinear pooling technique

4.3.2 MCB

The performance of the MCB fusion technique highly depends on its feature space dimensionality, which is required to be high-dimensional. Thus, we set this hyper-parameter as d_z = 16,000. Fig. 7 demonstrates a graphical map for the MCB operation, including all hyper-parameter values, all generated feature vectors step by step with their dimension values, and all weight matrices needed during fusion. Including the bias inputs, the number of learnable parameters for MCB is calculated as:

$$\text{Num.of MCB } parameters = size\ of\ W_z = (d_z + 1) \times l \tag{15}$$

Therefore, the fusion module in our Arabic-VQA system using the MCB fusion technique has just 32,002 learnable parameters.

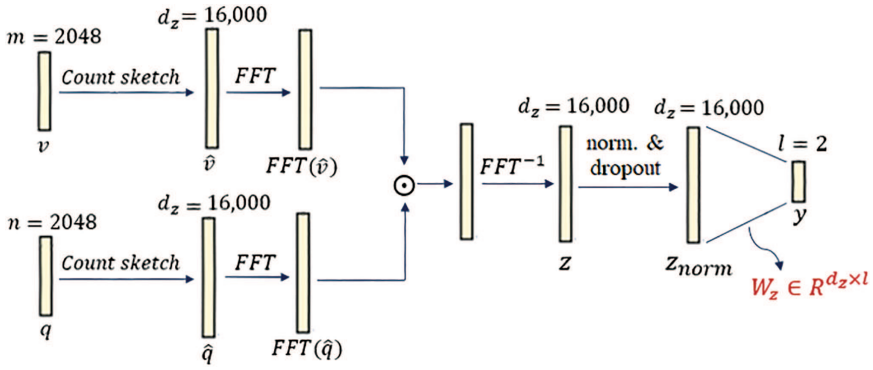

Figure 7: Graphical representation of the MCB fusion technique

4.3.3 MLB

Although the MLB fusion technique can generate a much lower-dimensional joint feature representation than MCB, it is still sensitive to its feature space dimensionality. Hence, in our system, the value of this hyper-parameter is set to d_z = 1200. Fig. 8 exhibits a graphical map of the MLB operation, including all hyper-parameter values, all generated feature vectors step by step with their dimension values, and all weight matrices needed during fusion. Including the bias inputs, the number of learnable parameters for MLB is

calculated as:

$$\begin{aligned}\text{Num.of MLB } parameters &= size\,of\ W_v + size\,of\ W_q + size\,of\ W_z \\ &= (m+1)\times d_z + (n+1)\times d_z + (d_z+1)\times l\end{aligned} \tag{16}$$

Thus, there are 4,920,002 learnable parameters just for the fusion module in our Arabic-VQA system using the MLB fusion technique.

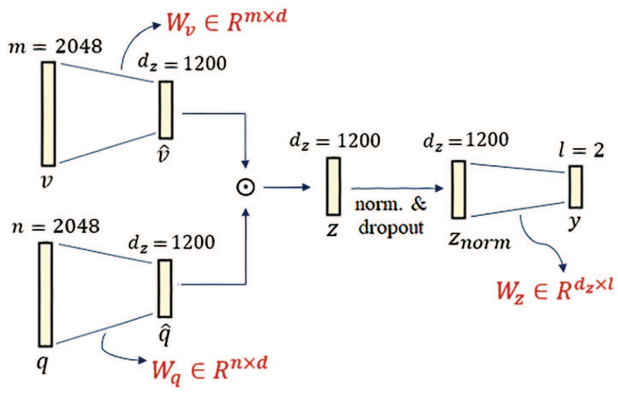

Figure 8: Graphical representation of the MLB fusion technique

4.3.4 MFB

As previously discussed in Section 2, the values of the two hyper-parameters k and o highly impact the VQA model performance and the model complexity as well. For this reason, the values of these hyper-parameters are defined as $k = 5$ and $o = 1000$. Fig. 9 graphically represents the MFB fusion technique with its two phases of operation, including all hyper-parameter values, all generated feature vectors step by step with their dimension values, and all weight matrices needed during fusion. Including the bias inputs, the number of learnable parameters for MFB is calculated as:

$$\begin{aligned}\text{Num.of MFB } parameters &= size\,of\ W_v + size\,of\ W_q + size\,of\ W_z \\ &= (m+1)\times k\times o + (n+1)\times k\times o + (o+1)\times l\end{aligned} \tag{17}$$

Subsequently, the fusion module in our Arabic-VQA system using the MFB technique contains 20,492,002 learnable parameters.

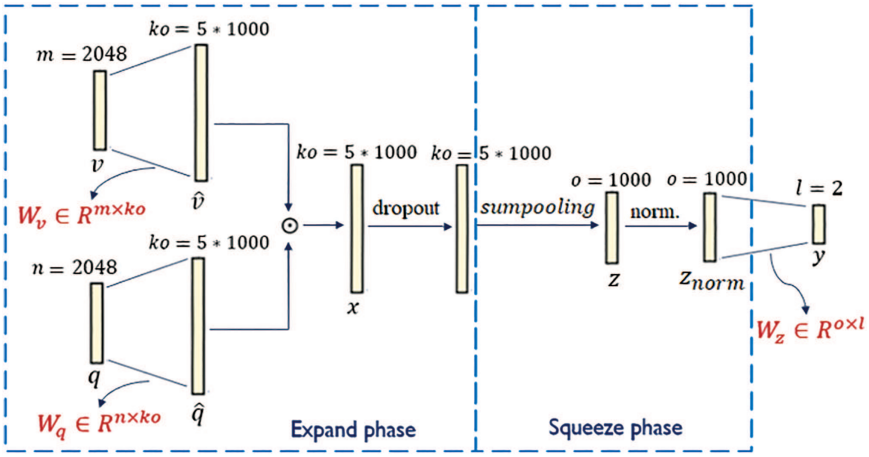

Figure 9: Graphical representation of the MFB fusion technique

4.3.5 MFH

The MFH technique has been utilized in our system so that only two MFB blocks are cascaded, where $p = 2$. This is because the authors in [8] have found that MFH_3 (i.e., cascading three MFB blocks) has a slightly worse performance than MFH_2, other than increasing the model complexity. We have used the same values of $k = 5$ and $o = 1000$ as we have done in the MFB technique. Fig. 10 illustrates the general block diagram of the MFH fusion technique, where in our system the MFB block is repeated only two times. Including the bias inputs, the number of learnable parameters for MFB is calculated as:

$$\begin{aligned} \text{Num.of MFH } parameters &= p \times (size\ of\ W_v + size\ of\ W_q) + size\ of\ W_z \\ &= p \times [(m+1) \times k \times o + (n+1) \times k \times o] + (p \times o + 1) \times l \\ &\approx p \times number\ of \text{ MFB } parameters \end{aligned} \tag{18}$$

So, the number of learnable parameters for the fusion module in our Arabic-VQA system using the MFH technique is 40,984,002 parameters.

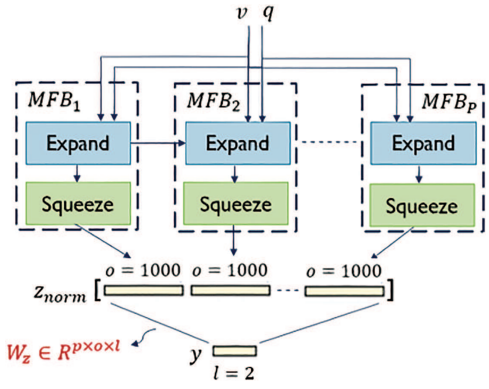

Figure 10: Block diagram of the MFH fusion technique

4.3.6 MUTAN

Both versions of the MUTAN technique are utilized in our Arabic-VQA system. For the first version, the three dimensions d_v, d_q, and d_z represent a stumbling block between the VQA model robustness and the model complexity. So, the values of these hyper-parameters are set to $d_v = d_q = d_z = 160$. Fig. 11a exhibits a graphical representation of the first version of the MUTAN technique, where no structural constraints are enforced on the 3-D core matrix T_c. Including the bias inputs, the number of parameters for this version of the MUTAN technique is calculated as:

$$\begin{aligned} &\text{Num. of } parameters\ for \text{ MUTAN\_without\_Rank} \\ &\quad = size\ of\ W_v + size\ of\ W_q + size\ of\ \mathcal{T}_c + size\ of\ W_z \\ &\quad = (m+1) \times d_v + (n+1) \times d_q + (d_v \times d_q + 1) \times d_z + (d_z + 1) \times l \end{aligned} \tag{19}$$

Thus, the number of learnable parameters contained in the fusion module in our Arabic-VQA system using the MUTAN technique without enforcing a rank constraint on the core matrix is 4,752,162 parameters.

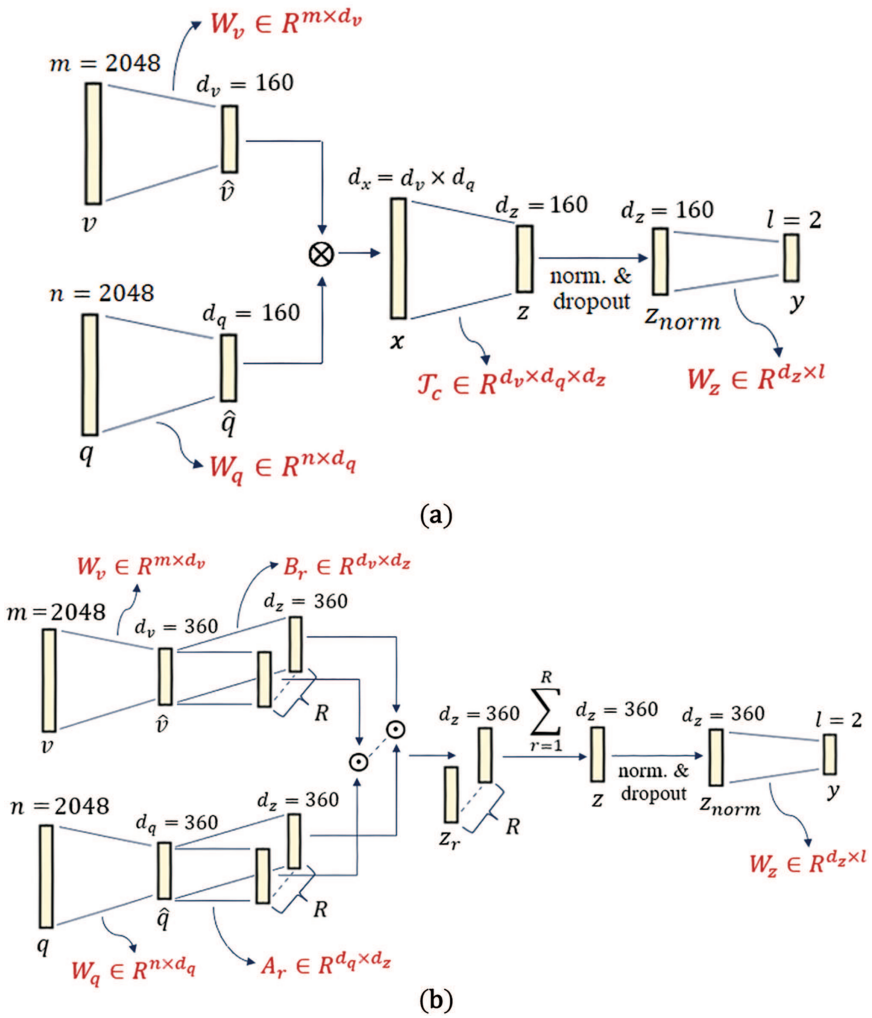

Figure 11: Graphical representation for MUTAN fusion (a) without and (b) with enforcing a fixed rank constraint on the core matrix

In the second version, the values for the three dimensions are allowed to be slightly increased, while an additional hyper-parameter R is enforced. The values of the three dimensions and the enforced rank are set to $d_v = d_q = d_z = 360$ and $R = 10$. Fig. 11b graphically represents the MUTAN operation while introducing a fixed rank constraint on the core matrices. Including the bias inputs, the number of MUTAN parameters in this case is defined as:

$$\begin{aligned}&\text{Num. of } parameters\,for \text{ MUTAN\_with\_Rank}\\&\quad = size\,of\ W_v + size\,of\ W_q + R \times (size\,of\ A_r + size\,of\ B_r) + size\,of\ W_z\\&\quad = (m+1) \times d_v + (n+1) \times d_q + R \times [(d_q+1) \times d_z + (d_v+1) \times d_z] + (d_z+1) \times l\end{aligned} \tag{20}$$

Hence, there are 4,075,202 learnable parameters present in the fusion module in our Arabic-VQA system using the MUTAN technique while a fixed rank constraint is enforced on the core matrix.

4.3.7 BLOCK

Just like MUTAN, we have utilized both versions of the BLOCK technique in our Arabic-VQA system. For the first version, the hyper-parameters are set as $k_v = k_q = k_z = 80$ and the number of chunks $C = 20$. Fig. 12a shows a graphical representation of the BLOCK technique without enforcing a fixed rank constraint to the core blocks D_c. Including the bias inputs, the number of parameters for BLOCK without enforcing a fixed rank constraint is calculated as:

$$\begin{aligned}&\text{Num. of } parameters\,for\,BLOCK\text{\_without\_Rank}\\&\quad = size\,of\ W_v + size\,of\ W_q + C \times size\,of\ D_c + size\,of\ W_z\\&\quad = (m+1) \times Ck_v + (n+1) \times Ck_q + C \times (k_v \times k_q + 1) \times k_z + (Ck_z + 1) \times l\end{aligned} \tag{21}$$

Subsequently, the total number of learnable parameters for the fusion module in our Arabic-VQA system using the BLOCK technique without enforcing a rank constraint on the core blocks is 16,801,602 parameters.

Like the first version of the BLOCK technique, the values of $k_v = k_q = k_z = 80$ and $C = 20$ are adopted for the second version, while the enforced rank is set to $R = 10$. Fig. 12b graphically represents the BLOCK fusion technique while enforcing a fixed rank constraint to each core block D_c. Including the bias inputs, the number of BLOCK parameters in this case is calculated as:

$$\begin{aligned}&\text{Num. of } parameters\,for\,BLOCK\text{\_with\_Rank}\\&\quad = size\,of\ W_v + size\,of\ W_q + C \times R \times (size\,of\ A_r^c + size\,of\ B_r^c) + size\,of\ W_z\\&\quad = (m+1) \times Ck_v + (n+1) \times Ck_q + C \times R \times [(k_q+1) \times k_z + (k_v+1) \times k_z] + (Ck_z+1) \times l\end{aligned} \tag{22}$$

So, there are 9,152,002 learnable parameters present in the fusion module in our Arabic-VQA system using the BLOCK technique while a fixed rank constraint is enforced on the core blocks.

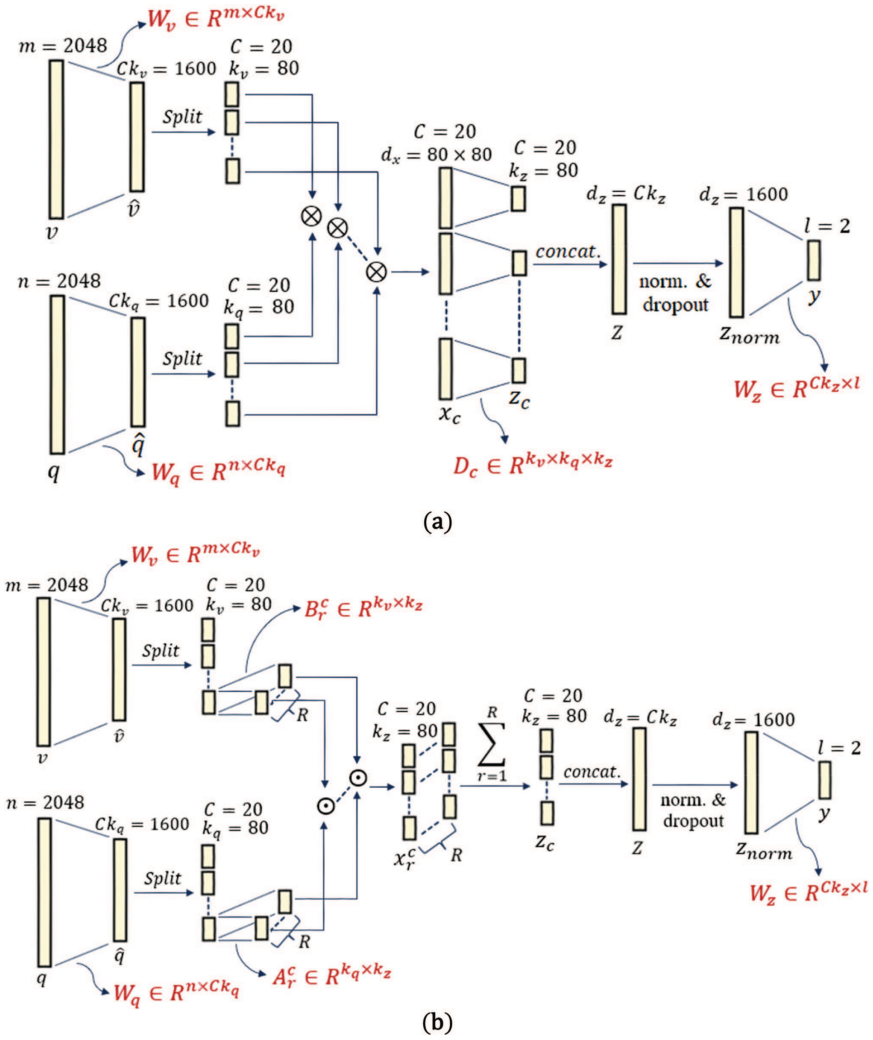

Figure 12: Graphical representation for BLOCK fusion (a) without and (b) with enforcing a fixed rank constraint on the core matrices

4.3.8 MLPB

As described in Section 3, the MLPB technique is composed of two phases, which are the question-kernel-based pooling phase and the image-kernel-based pooling phase. In the question-kernel-based pooling phase, the question kernel should slide over the image embedding with a pre-defined stride s, where T_q is the total number of sliding steps needed for passing the question kernel over the entire image embedding, and *vice versa* for the image-kernel-based pooling phase. In our system, we have assumed a non-overlapped window to slide the question kernel over the image embedding in the question-kernel-based pooling phase, and *vice versa* for the image-kernel-based pooling phase. This is for reducing the model complexity. Hence, we set the dimensions of both image and question kernels and their stride values as $p = s = 128$. Thus, in the question-kernel-based pooling phase, the image embedding is divided into $T_q = 16$ non-overlapped clips. Similarly, in the image-kernel-based pooling phase, the question embedding is divided into $T_v = 16$ non-overlapped clips. The output feature space dimensionality for each sliding step (t) is set as $o = 100$.

Fig. 13a shows a graphical representation of the question-kernel-based pooling phase of MLPB, while Fig. 13b illustrates a block diagram for the entire MLPB fusion technique, including all hyper-parameter

values, all generated feature vectors step by step with their dimension values, and all weight matrices needed during fusion. Including the bias inputs, the number of learnable parameters for the question-kernel-based pooling phase of MLPB is calculated as:

$$\begin{aligned}\text{Num. of } parameters\ of\ q_k \text{ phase of MLPB (unshared\_parameters)} &= size\ of\ W_q + T_q \times \left(size\ of\ W_{x_q}^t\right) \\ &= (n+1) \times p + T_q \times [(p \times p + 1) \times o]\end{aligned} \tag{23}$$

To minimize the number of phase parameters, the authors in [9] have suggested sharing the learning parameters for all steps (i.e., all the product results x_q^t of all steps pass through the same fully connected layer). So, all the weight matrices $W_{x_q}^1, \ldots, W_{x_q}^t, \ldots, W_{x_q}^{T_q}$ are replaced by just one weight matrix W_{x_q}. Thus, the number of parameters for the question-kernel-based pooling phase is recalculated as:

$$\begin{aligned}\text{Num. of } parameters\ of\ q_k \text{ phase of MLPB (shared\_parameters)} &= size\ of\ W_q + size\ of\ W_{x_q} \\ &= (n+1) \times p + (p \times p + 1) \times o\end{aligned} \tag{24}$$

It is worth noting that the number of the image-kernel-based pooling phase parameters is also calculated in the same way as in Eq. (23) for the case of not sharing learning parameters, and as in Eq. (24) for the case of sharing learning parameters for all steps. The total number of parameters for MLPB including both phases is calculated as:

$$\begin{aligned}&\text{Num.of } parameters \text{ of MLPB} \\ &\quad = \text{Num.of } parameters\ of\ q_k \text{ phase} + \text{Num.of } parameters\ of\ v_k \text{ phase} + \text{size of } W_z \\ &\quad = \text{Num. of } parameters\ of\ q_k\ phase + \text{Num.of } parameters\ of\ v_k\ phase + (o \times T_q + o \\ &\qquad \times T_v + 1) \times l\end{aligned} \tag{25}$$

For the MLPB configuration, there are two options, either to adopt the sharing learning parameters concept or adopt the concept of non-sharing the learning parameters. In our system, we have adopted only the sharing parameters concept, in both phases of MLPB technique. This is because the total number of parameters for the fusion module is 3,807,946 parameters, in the case of sharing the learning parameters. In contrast, the total number of parameters for the fusion module is 52,962,946 parameters, in the case of non-sharing the learning parameters. This inflates the model's size dramatically. Further, the authors in [9] have found that the MLPB without sharing the parameters has a slightly worse performance than the MLPB with sharing the parameters, other than increasing the model complexity.

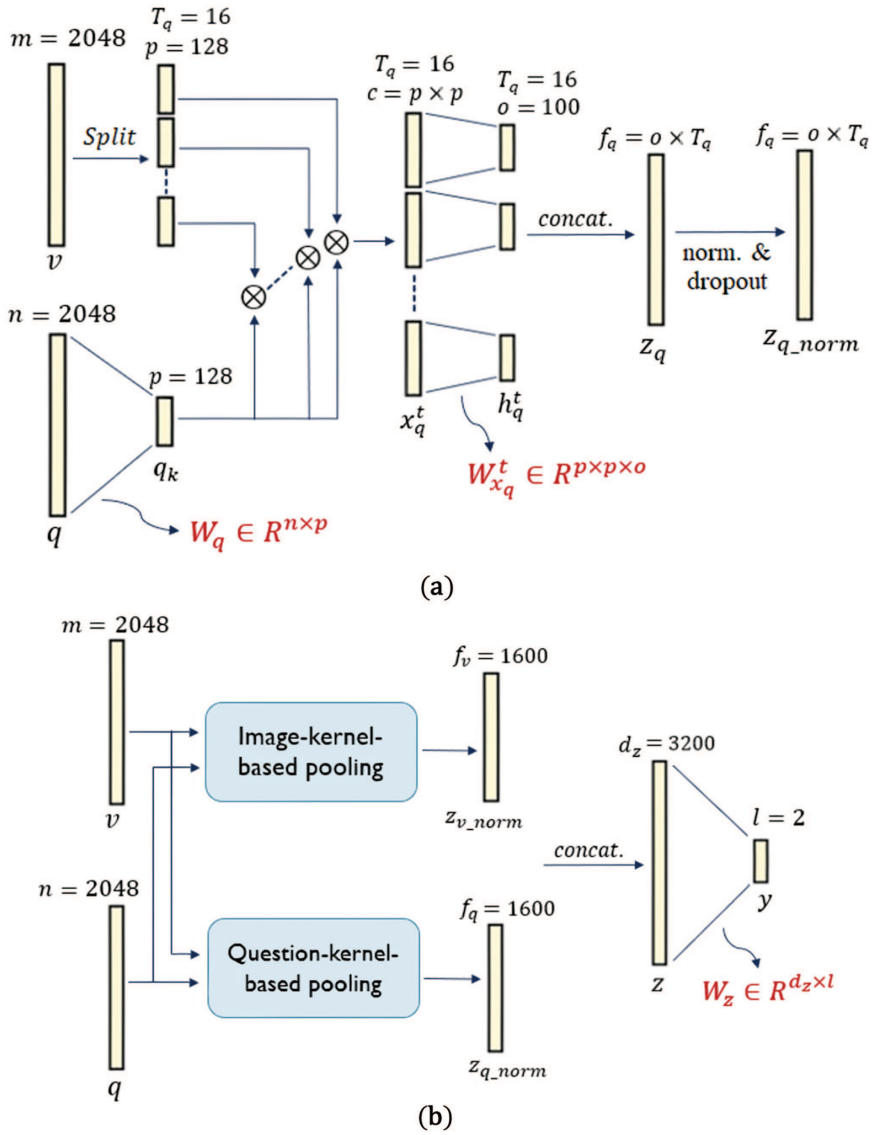

Figure 13: (a) graphical representation for the question-kernel-based pooling phase of MLPB, (b) block diagram for the entire MLPB fusion technique

4.4 Answer Prediction

This module is represented as a binary classification task over a closed set of two candidate answers "نعم" and "لا". By getting the joint embedding vector $y \in R^l$ from the feature fusion module, where $l = 2$ is the number of classes in the VQA model, the appropriate answer is simply predicted by applying the softmax function to calculate the probability distribution over the two answers. The predicted answer is determined as the class with the highest softmax probability, which is defined as follows:

$$ans = \arg\max(softmax(y)) \tag{26}$$

5 Experimental Results and Discussion

Experiments have been performed on Google Colaboratory [65], where free access is enabled to a virtual machine of a Nvidia T4s GPU with 12 GB RAM. All VQA models and bilinear pooling algorithms included in our experiments are implemented using the Python programming language along with PyTorch libraries.

All models are trained using the Negative Log Likelihood loss function, where parameters of the ResNet-152, and the SG word embedding model from Aravec 2.0 are kept frozen during training. Models are learned using the Adam optimizer for 40 epochs, with a batch size of 32. The learning rate is initially set to 1×10^{-3} and scheduled to be decreased by a multiplicative factor of 0.1 every 10 epochs.

All our Arabic-VQA models are developed on the VAQA dataset [25]. It is the first VQA dataset in Arabic, that is dedicated to yes/no questions about real-world images. The dataset is fully automatically generated, containing 5000 images taken from the MS-COCO dataset and 2712 Arabic questions, resulting in 137,888 Image-Question-Answer (IQA) triplets. The dataset is divided into 60%: 20%: 20% for training, testing, and validation, respectively. Each of the three splits contains a distinct set of images and a distinct set of questions. Questions in this dataset have three different tasks: COCO object existence verification, COCO super-category existence verification, and image scene recognition. Fig. 14 shows some samples of IQA triplets from the VAQA dataset. For non-Arabic native readers, the English translation of the input question and the output answer are provided within the same figure.

Our experiments on the Arabic-VQA system are performed using eight bilinear pooling fusion techniques, as discussed in Section 4. Further, MUTAN and BLOCK techniques have been utilized with and without enforcing a fixed-rank constraint on the core matrices. The two versions of MUTAN and BLOCK techniques are abbreviated as MUTAN_w_rank, MUTAN_wo_rank, BLOCK_w_rank, and BLOCK_wo_rank, respectively. This results in ten Arabic-VQA models, each developed using a different fusion technique. Whereas, approaches used in the remaining modules of question pre-processing, question feature extraction, image feature extraction, and answer prediction are retained the same for all the developed Arabic-VQA models. The accuracy, F1-score, precision, and recall evaluation metrics are utilized to assess the performance of our developed Arabic-VQA models. Table 4 exhibits a comparison between the ten Arabic-VQA models, where the comparison is performed in five aspects:

1. The dimensionality of the generated bilinear vector z from each fusion technique. This bilinear vector is the key factor, that all these bilinear pooling fusion techniques try to minimize its dimensionality to ultimately reduce the model complexity.
2. The model complexity is represented by the number of parameters for the feature fusion module. This is because the only difference between all these Arabic-VQA models is in the fusion technique utilized.
3. The performance of each Arabic-VQA model is represented by their scores achieved for the accuracy and F1-score evaluation metrics in Table 4. The precision and recall records for all these Arabic-VQA models are presented in Fig. 15.
4. The model size in memory.
5. The average inference time required for each input image-question pair.

Arabic questions with their answers	هل تتمكن من رؤية حيوان فى الصورة؟ نعم هل تحتوى هذه الصورة على دَرّاجة آليّة؟ لا أتعثر على زَرافة فى الصورة؟ لا أهنالك أُناس فى الصورة؟ نعم هل تم الحصول على تلك الصورة فى مكان مغلق؟ لا
English translation	Can you see an animal in the picture? Yes Does this picture contain a motorcycle? No Do you find a giraffe in the picture? No Are there people in the picture? Yes Was this picture taken indoors? No

(a)

Arabic questions with their answers	أيتسنى العثور على شَوْكة طَعام فى الصورة؟ نعم أتتضمن هذه الصورة مِحمَصة خُبْز كهربائية؟ لا هل هناك قِطار فى الصورة؟ لا هل تستطيع إيجاد طاولات طَعام فى الصورة؟ نعم أتجد مُعدّات رياضية فى الصورة؟ لا
English translation	Can a fork be found in the picture? Yes Does this picture include a toaster? No Is there a train in the picture? No Can you find dining tables in the picture? Yes Do you find any sports equipment in the picture? No

(b)

Figure 14: Samples of image-question-answer triplets from the VAQA dataset

Table 4: Comparison between all the developed Arabic-VQA models, in terms of the dimensionality of the generated bilinear vector from each fusion technique, the model complexity and its size, their performance on the VAQA dataset, and the average inference time for each model

Image features extraction	Question features extraction	Feature fusion	Dimensionality of the bilinear vector z	Num. of parameters for fusion module	Model size (MB)	Accuracy %	F1-Score	Average inference time (s)
ResNet-152	1-layer GRU	FBP	4,194,304	8,388,610	261.2	87.606	0.876	0.0066
		MCB	16,000	32,002	165.7	89.048	0.886	0.0064
		MLB	1200	4,920,002	221.6	89.051	0.888	0.0063
		MFB	1000	20,492,002	399.8	89.055	0.889	0.0065
		MFH	2000	40,984,002	634.4	89.259	0.891	0.0068
		MUTAN_wo_rank	160	4,752,162	216.9	88.540	0.884	0.0063
		MUTAN_w_rank	360	4,075,202	211.9	87.781	0.877	0.0065
		BLOCK_wo_rank	1600	16,801,602	357.6	88.500	0.883	0.0066
		BLOCK_w_rank	1600	9,152,002	270.1	88.174	0.880	0.0065
		MLPB	3200	3,807,946	208.8	89.167	0.892	0.0064

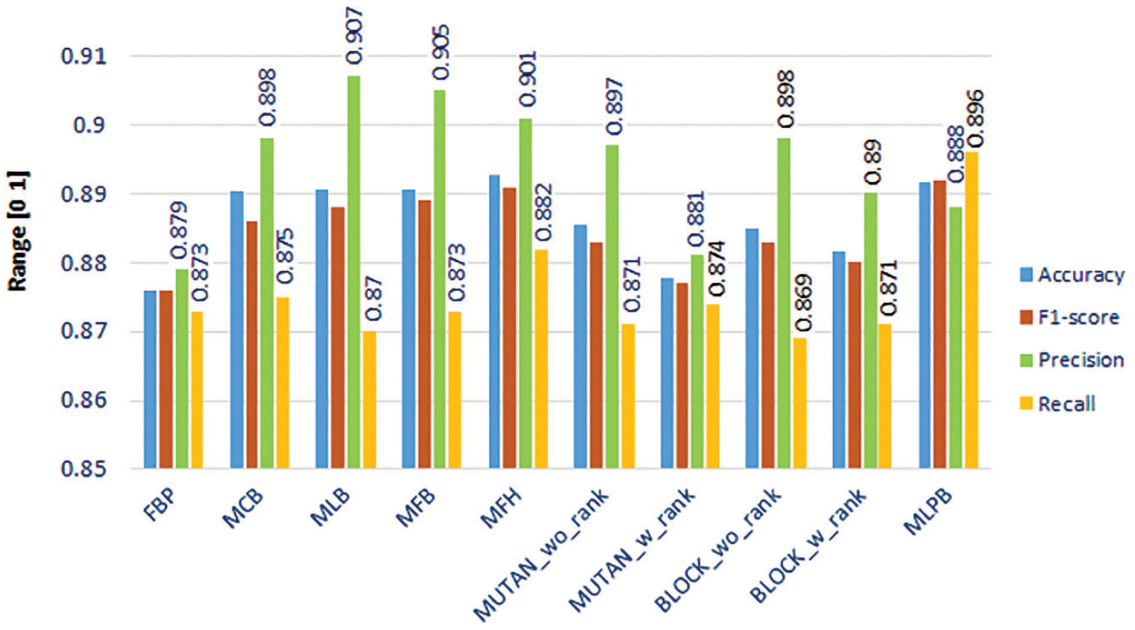

Figure 15: A chart-based performance comparison between all the Arabic-VQA models, each developed using a different bilinear pooling fusion technique

From Table 4, we can observe that all these bilinear pooling fusion techniques have improved the VQA model performance from 87.6% until reaching 89.25%, compared to the FBP technique as a reference. More precisely, the MCB, MLB, MFB, MFH, and MLPB techniques outperform all the other bilinear pooling fusion techniques, exceeding 89%. Fig. 15 exhibits a chart-based performance comparison between all these Arabic-VQA models, in terms of their scores achieved for the accuracy, F1-score, precision, and recall evaluation metrics where the accuracy is scaled in the range [0, 1] to be represented with the other three metrics. It can be noticed that the MFH has achieved the highest accuracy, while the MLB has achieved the highest precision. The MLPB has achieved the second-highest accuracy after the MFH while achieving the highest F1 score and recall values. In terms of inference time, all these techniques take approximately the same amount of time. This is because these ten Arabic-VQA models utilize the same approaches in their modules, except in the fusion module which slightly affected the average inference time.

Regarding the number of learnable parameters, the inapplicability of the fully parameterized bilinear pooling technique can't be considered a general case for all VQA systems. It can be applied to VQA systems designed to answer yes/no questions, where there are two candidate answers $l = 2$. This is because the number of parameters is just 8 million, not billions as in the VQA systems with thousands of answers. It requires fewer parameters than some other bilinear pooling fusion techniques, as demonstrated in Table 4.

Although the Arabic-VQA model that utilized the MFH fusion technique has achieved the highest accuracy of 89.25%, it also has the largest number of learnable parameters. The MFH technique has increased the number of parameters to 40.9 million parameters, indicating the highest model complexity among all bilinear pooling fusion techniques. Compared to the FBP technique as a reference, MFB has increased the number of parameters to 20.4 million parameters. Similarly, the two versions of BLOCK have increased the number of parameters to reach 9 and 16.8 million parameters. Hence, the BLOCK (with both versions), MFB, and MFH techniques are not recommended for VQA systems intended to answer yes/no questions. This is

because these techniques can't accomplish their main objective of reducing the model complexity for this case of VQA systems. Instead, they have significantly increased the number of learnable parameters.

On the other hand, MCB, MLB, MLPB, and MUTAN (with both versions) have efficiently reduced the number of learnable parameters, compared to the FBP technique as a reference. Instead of 8 million parameters in the FBP, the MLB has 4.9 million parameters, the MLPB has 3.8 million parameters, the MCB has just 32 thousand parameters, and the two versions of MUTAN have 4 and 4.7 million parameters. Thus, we can say that their main goal of reducing the model complexity is accomplished whether the developed VQA model has a small number of answers (as proven in this work) or a large number of answers (as proven in their published articles).

All these bilinear pooling fusion techniques have considered that the dimensionality of the bilinear vector z in the FBP is the main reason that causes its huge parameter space issue, as the dimensionality of this bilinear vector after the outer product operation is usually in millions. This leads to several billion learnable parameters just in the feature fusion module, as all these techniques have been examined only on VQA systems with thousands of answers. Therefore, these techniques tend to add multiple projection layers during the fusion process, to project the input embeddings into much lower dimensional spaces before generating this bilinear vector z. This is in hopes of reducing the dimensionality of the resultant bilinear vector z as much as possible and hence reducing the size of the last weight matrix before generating the appropriate answer.

In contrast, the dimensionality of the bilinear vector z is not the only reason for the parameter space issue for the VQA systems specialized in yes/no questions, as the size of this bilinear vector will be multiplied by a small number of classes (i.e., $l = 2$). Thus, the different attempts of these techniques to reduce the dimensionality of this bilinear vector by adding multiple and large-sized projection layers during the fusion process will be the main reason for the massive parameter space issue. This is because adding all these fully connected layers could lead to more parameters than in the case of producing a high-dimensional bilinear vector z. That is why MFB, MFH, and BLOCK have increased the number of model parameters. It is also the reason why MCB has the least number of learnable parameters among all bilinear pooling fusion techniques, although it produces the second largest bilinear vector z of dimensionality $d = 16{,}000$.

On the other hand, some related studies have considered that MCB can't sufficiently solve the huge parameter space issue of bilinear pooling, as they examined it only for VQA systems with thousands of answers. For example, suppose a VQA model sets $d = 16{,}000$ and has a set of answers of size $l = 3000$. The number of learnable parameters for the fusion module will exceed 48 million parameters. Hence, the number of answers in the VQA systems is a critical factor that affects the model complexity, which also affects the ability of all these bilinear pooling fusion techniques to reduce this complexity. Therefore, we can give recommendations for all these fusion techniques according to the number of answers in the developed VQA systems, as shown in Table 5.

Table 5: Recommendations for multimodal bilinear pooling fusion techniques, according to the number of answers in the VQA systems

Number of answers in VQA systems	FBP	MCB	MLB	MFB	MFH	MUTAN	BLOCK	MLPB
A large number of answers	✗	✓	✓	✓	✓	✓	✓	✓
A small number of answers	✓	✓	✓	✗	✗	✓	✗	✓

Fig. 16 illuminates a chart-based comparison between all these bilinear pooling fusion techniques, in terms of the achieved model performance and the number of their learnable parameters for VQA models dedicated to yes/no questions. It can be noticed that the MLPB, MLB, and MCB have shown good balances of high models' performance with low models' complexity. Specifically, the MLPB technique has proven the best balance in the trade-off between the VQA model complexity and the overall model performance, for Arabic-VQA models designed to answer yes/no questions. This is because it ranks the second-highest accuracy after MFH and the highest F1 score and recall while ranking the second-lowest complexity after MCB. It has achieved a performance of 89.16% which is very close to the best performance of 89.25% achieved by MFH, while having just 3.8 million parameters instead of 40.9 million parameters.

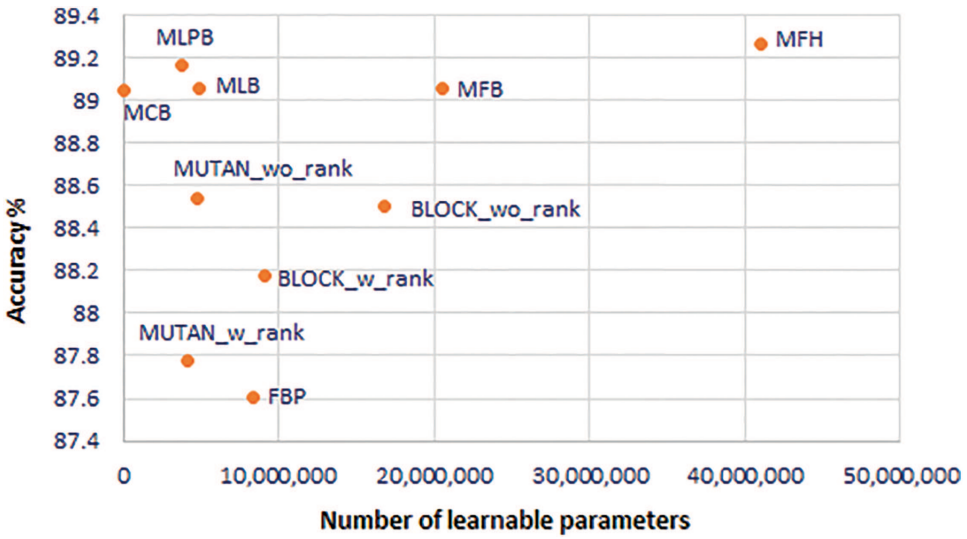

Figure 16: Performance vs. complexity chart-based comparison between all multimodal bilinear pooling fusion techniques, for Arabic-VQA models dedicated to yes/no questions

5.1 Comparison with the State-of-the-Art

Table 6 presents a comparison between our best Arabic-VQA model using the MLPB fusion technique and the Arabic-VQA model proposed in [25], in terms of the model performance achieved on answering Arabic yes/no questions. This Arabic-VQA model [25] has been developed on the same VAQA dataset with a similar framework. This comparison is conducted to explore how the different techniques and approaches utilized in our Arabic-VQA model contribute to improving the model performance from 84.936% to 89.167%. Hence, three additional experiments are performed, where the three different techniques used in our proposed Arabic-VQA model during the image features extraction, question features extraction, and feature fusion modules are added one by one to the Arabic-VQA model proposed in [25]. This aims to investigate the effect of adding each of these techniques on the overall model performance. These experiments are described as follows:

1. In the first experiment, all modules of the Arabic-VQA model in [25] remain the same, except that the MLPB fusion technique is utilized instead of the element-wise multiplication technique used in [25]. This is to determine whether the MLPB technique can improve the model performance on its own or not.

2. In the second experiment, we examined the contribution of adding ResNet-152 along with the MLPB fusion technique to the model performance, while the question feature extraction module has remained the same as in [25].
3. In the third experiment, the GRU is added along with the MLPB fusion technique to explore their effect on the model performance, while the image feature extraction module has remained the same as in [25].
4. In the fourth experiment, we investigated how ResNet-152 and GRU contribute to the model performance on their own without using the MLPB fusion technique. Hence, the ResNet-152 and GRU are used along with the element-wise multiplication technique that has been used in [25].

Table 6: Comparison between our proposed Arabic-VQA system and the state-of-the-art on the VAQA dataset

VQA model	Image representation	Question representation	Feature fusion	Accuracy %	F1 score
Kamel et al. [25]	VGG-16	1-layer Uni-LSTM	Element-wise multiplication	84.936	Not specified
Proposed experiment 1	VGG-16	1-layer Uni-LSTM	MLPB	87.928	0.878
Proposed experiment 2	ResNet-152	1-layer Uni-LSTM	MLPB	88.807	0.889
Proposed experiment 3	VGG-16	1-layer Uni-GRU	MLPB	88.590	0.884
Proposed experiment 4	ResNet-152	1-layer Uni-GRU	Element-wise multiplication	87.257	0.871
Our proposed system	ResNet-152	1-layer Uni-GRU	MLPB	89.167	0.892

From Table 6, it can be noticed that replacing the element-wise multiplication with the MLPB fusion technique in the first experiment improved the model performance from 84.93% to 87.92%. Moreover, the addition of ResNet-152 with MLPB in the second experiment has further improved the model performance to 88.8%. Similarly, the addition of GRU with MLPB in the third experiment has further improved the model performance to 88.59%. On the other hand, in the fourth experiment, the use of ResNet-152 with GRU improved performance from 84.93% to 87.25%, compared to the model in [25] where element-wise multiplication was used as well. Hence, the usage of both ResNet-152 and GRU with MLPB in the proposed Arabic-VQA model has upgraded the model performance to achieve 89.167%.

Table 7 illustrates a comparison between the performance of our Arabic-VQA system developed on the VAQA dataset with related VQA systems, considering only their performance on yes/no questions according to the scope of our work. Although these related VQA systems were constructed using other VQA datasets with other natural languages, they have utilized deep learning approaches for both image and question representation modules and used the category of multimodal bilinear pooling fusion techniques, as in our proposed Arabic-VQA system.

In Table 7, it can be noticed that both VQA models in [6,12] have utilized the same ResNet-152 and GRU techniques along with one of the multimodal bilinear pooling fusion techniques, similar to our proposed Arabic-VQA system. In Table 4, it is proven that the MLPB outperforms both MLB and MUTAN techniques, for VQA models that are developed on the same VAQA dataset. Similarly, in Table 7, our Arabic-VQA

system using the MLPB technique outperforms both VQA models in [6,12] that used the MLB and MUTAN techniques, respectively. We can say that the achieved performance of our proposed Arabic-VQA system is good and very comparable to the performance of related VQA models for the same type of questions. Further, both "نعم" and "لا" answers in the utilized VAQA dataset have balanced distributions [25]. This enforces the developed VQA models to learn properly by preventing cheating, where models can rely on priors and most frequent answers in biased datasets to predict a potential answer without reasoning [66–68]. Hence, this supports the model generalization.

Table 7: Comparison between our proposed Arabic-VQA system and related works that utilized approaches from similar categories through their pipelines, in terms of their performance only for yes/no questions

VQA research ref.	Image representation	Question representation	Feature fusion	VQA dataset	Language	Accuracy %
Kim et al. [6]	ResNet-152	GRU	MLB	VQA1.0 [21]	English	84.61
Ben-younes et al. [12]	ResNet-152	GRU	MUTAN	VQA1.0 [21]	English	85.14
Fukui et al. [11]	ResNet-152	LSTM	MCB	VQA1.0 [21]	English	83.4
Yu et al. [7]	ResNet-152	LSTM	MFB	VQA1.0 [21]	English	85.6
Yu et al. [8]	ResNet-152	LSTM	MFH	VQA1.0 [21]	English	86.2
Lao et al. [9]	ResNet	LSTM	MLPB	VQA2.0 [66]	English	80.57
Ben-younes et al. [13]	Faster R-CNN	GRU	BLOCK	VQA2.0 [66]	English	82.86
Su et al. [35]	ResNet-152	LSTM	MLB	VQA1.0 [21]	English	84.1
				VQA2.0 [66]	English	83.70
Osman et al. [34]	Faster R-CNN	LSTM	MCB	VQA1.0 [21]	English	84.92
				VQA2.0 [66]	English	82.85
Shuang et al. [45]	Faster R-CNN	GRU	MUTAN	VQA2.0 [66]	English	90.36
Miao et al. [50]	Faster R-CNN	GRU	BLOCK	VQA2.0 [66]	English	84.23
Mohamud et al. [51]	Faster R-CNN	GRU	BLOCK	VQA2.0 [66]	English	83.98
Our proposed system	ResNet-152	GRU	MLPB	VAQA [25]	Arabic	89.167

Table 8 presents another comparison between the performance of our Arabic-VQA system developed on the VAQA dataset with related transformer-based VQA systems. These VQA models have been developed on other VQA datasets in different domains, such as medical-VQA, remote sensing VQA, and the general VQA task as well. Performance on yes/no questions only is considered according to our scope of work.

Although Table 8 provides a subjective comparison with VQA models that used different techniques and developed on different datasets, it has shown that the performance of our proposed Arabic-VQA system is comparable to the performance of these transformer-based VQA models for the same type of questions. However, the use of vision and language transformers could benefit the robustness of Arabic-VQA systems, which could be investigated in the future.

Table 8: Comparison between our proposed Arabic-VQA system and related transformer-based VQA systems, in terms of their performance only for yes/no questions

VQA research ref.	Utilized techniques	VQA dataset	Language	Accuracy %
Yan et al. [31]	Faster-RCNN + Bert + BAN + EWM	VQA2.0 [66]	English	87.81
Xia et al. [69]	Faster-RCNN + LSTM + Shrinkage transformer	VQA2.0 [66]	English	87.33
Hackel et al. [32]	XCiT_Nano + BERT_TINY + EWM	RSVQAxBEN [70]	English	89.72
Khare et al. [55]	ResNet-152 + MMBERT	VQA-Med 2019 [71]	English	87.5
Silva et al. [56]	EfficientNetV2 + Real-Former Transformer	RSVQAxBEN [70]	English	87.57
Siebert et al. [58]	ResNet-152 + VisualBERT	RSVQAxBEN [70]	English	86.56
Our proposed system	ResNet-152 + GRU + MLPB	VAQA [25]	Arabic	89.167

Most VQA studies and datasets have focused on the English language, where the VQA1.0 [21] and VQA2.0 [66] are the benchmark datasets for most studies. Recently, some studies have proposed multilingual VQA systems and datasets, as in [72–76]. These studies have adopted translation-based approaches for multilingual-VQA dataset generation. However, none of these studies have considered the Arabic language. Furthermore, none of these VQA systems explicitly investigated for yes/no questions.

6 Conclusion and Future Work

In this paper, we have proposed a VQA system for answering yes/no questions about real-world images, in Arabic. The proposed Arabic-VQA system consists of five modules, all of which are developed on deep learning approaches. These modules are image features extraction, question pre-processing, question features extraction, feature fusion, and answer prediction. A ResNet-152 model has been employed for image representation. A one-layer unidirectional GRU has been adopted for semantically representing the input question. For feature fusion, eight multimodal bilinear pooling fusion techniques have been utilized, including FBP, MCB, MLB, MFB, MFH, MUTAN, BLOCK, and MLPB. Lastly, answers have been predicted using a softmax-based classifier.

All multimodal bilinear pooling fusion techniques were originally designed to reduce the model complexity as much as possible while preserving the model performance. In literature, there is a severe lack of validating the efficiency of these fusion techniques for VQA systems dedicated to yes/no questions where there are two candidate answers. Therefore, this study targets this case of VQA systems, to spotlight the high impact of the number of answers in VQA systems on the effectiveness of these fusion techniques in achieving their main objective of reducing the model complexity. In this work, we have answered three research questions about applying these multimodal bilinear pooling fusion techniques for this case of VQA systems, which are: (a) Will the FBP technique remain inapplicable due to the number of its learnable parameters? (b) Will all these bilinear pooling fusion techniques accomplish their main objective of reducing

the model complexity? (c) Will using these bilinear pooling fusion techniques improve the overall model performance?

After conducting experiments, we found that all these multimodal bilinear pooling fusion techniques have improved the VQA model performance from 87.6% to reach the best performance of 89.25%, compared to the FBP technique as a reference. More precisely, MFH, MFB, MLPB, MLB, and MCB outperform all the other bilinear pooling fusion techniques, exceeding 89%.

Regarding the model complexity, the inapplicability of the FBP technique can't be considered a general case for all VQA systems. This could be true for the VQA systems with a large number of answers, but not for VQA systems intended to answer yes/no questions. The MCB, MLB, MLPB, and MUTAN techniques have efficiently reduced the number of model parameters from 8.3 million parameters until reaching 32 thousand parameters, compared to the FBP technique as a reference. In contrast, MFB, MFH, and BLOCK techniques are not recommended for VQA systems dedicated to yes/no questions, as they can't accomplish their main objective of decreasing the model complexity. Instead, they have significantly increased the number of model parameters from 8.3 million parameters until reaching 40.9 million parameters, compared to the FBP technique as a reference.

Concerning the performance vs. complexity trade-off, the MLPB, MLB, and MCB have shown good balances of high models' performance with low models' complexity. Specifically, the MLPB technique has proven the best balance for VQA systems designed to answer yes/no questions. It has ranked the second-highest accuracy after MFH and the highest F1 score and recall while ranking the second-lowest complexity after MCB.

The dimensionality of the resultant bilinear vector was the only focus of all these multimodal bilinear pooling fusion techniques, aiming to minimize it as much as possible to tackle the huge parameter space issue of bilinear pooling fusion techniques. However, we can say that the number of answers in the VQA systems is another critical factor that affects the model complexity and the ability of all these bilinear pooling fusion techniques to reduce this complexity. Hence, in this work, several recommendations of these multimodal bilinear pooling fusion techniques have been proposed for future VQA systems according to their number of answers.

In the future, we will investigate other question types and more answers for the VQA task in Arabic. Further, we aim to study the impact of utilizing vision and language transformers with multi-head attention mechanisms on the robustness of Arabic-VQA systems.

Acknowledgement: Not applicable.

Funding Statement: The authors did not receive any funding for this study.

Author Contributions: Conceptualization, Sarah M. Kamel, Mai A. Fadel, Lamiaa Elrefaei, and Shimaa I. Hassan; Methodology, Sarah M. Kamel, Mai A. Fadel, Lamiaa Elrefaei, and Shimaa I. Hassan; Software, Sarah M. Kamel; Validation, Sarah M. Kamel, Mai A. Fadel, Lamiaa Elrefaei, and Shimaa I. Hassan; Formal analysis, Sarah M. Kamel, Mai A. Fadel, Lamiaa Elrefaei, and Shimaa I. Hassan; Investigation, Sarah M. Kamel, Mai A. Fadel, Lamiaa Elrefaei, and Shimaa I. Hassan; Writing—original draft preparation, Sarah M. Kamel; Writing—review and editing, Sarah M. Kamel, Mai A. Fadel, Lamiaa Elrefaei, and Shimaa I. Hassan; Visualization, Sarah M. Kamel; Supervision, Lamiaa Elrefaei and Shimaa I. Hassan; Project administration, Lamiaa Elrefaei. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, S.M.K., upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Malinowski M, Fritz M. A multi-world approach to question answering about real-world scenes based on uncertain input. In: Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS); 2014 Dec 8–13; Montreal, QC, Canada. p. 1682–90.
2. Zitnick CL, Agrawal A, Antol S, Mitchell M, Batra D, Parikh D. Measuring machine intelligence through visual question answering. AI Mag. 2016;37(1):63–72. doi:10.1609/aimag.v37i1.2647.
3. Ishmam MF, Shovon MSH, Mridha MF, Dey N. From image to language: a critical analysis of Visual Question Answering (VQA) approaches, challenges, and opportunities. Inf Fusion. 2024;106:102270. doi:10.1016/j.inffus.2024.102270.
4. Zhang D, Cao R, Wu S. Information fusion in visual question answering: a survey. Inf Fusion. 2019;52:268–80. doi:10.1016/j.inffus.2019.03.005.
5. Farazi M, Khan S, Barnes N. Accuracy vs. complexity: a trade-off in visual question answering models. Pattern Recognit. 2021;120:108106. doi:10.1016/j.patcog.2021.108106.
6. Kim JH, On KW, Lim W, Kim J, Ha JW, Zhang BT. Hadamard product for low-rank bilinear pooling. In: Proceedings of the International Conference on Learning Representations (ICLR); 2017 Apr 24–26; Toulon, France.
7. Yu Z, Yu J, Fan J, Tao D. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2017 Oct 22–29; Venice, Italy. p. 1839–48. doi:10.1109/ICCV.2017.202.
8. Yu Z, Yu J, Xiang C, Fan J, Tao D. Beyond bilinear: generalized multimodal factorized high-order pooling for visual question answering. IEEE Trans Neural Netw Learn Syst. 2018;29(12):5947–59. doi:10.1109/TNNLS.2018.2817340.
9. Lao M, Guo Y, Wang H, Zhang X. Multimodal local perception bilinear pooling for visual question answering. IEEE Access. 2018;6:57923–32. doi:10.1109/ACCESS.2018.2873570.
10. Tenenbaum JB, Freeman WT. Separating style and content with bilinear models. Neural Comput. 2000;12(6):1247–83. doi:10.1162/089976600300015349.
11. Fukui A, Park DH, Yang D, Rohrbach A, Darrell T, Rohrbach M. Multimodal compact bilinear pooling for visual question answering and visual grounding. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2016 Nov 1–5; Austin, TX, USA. p. 457–68. doi:10.18653/v1/d16-1044.
12. Ben-younes H, Cadene R, Cord M, Thome N. MUTAN: multimodal tucker fusion for visual question answering. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2017 Oct 22–29; Venice, Italy. p. 2631–9. doi:10.1109/ICCV.2017.285.
13. Ben-younes H, Cadene R, Thome N, Cord M. BLOCK: bilinear superdiagonal fusion for visual question answering and visual relationship detection. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI); 2019 Jan 27–Feb 1; Honolulu, HI, USA. p. 8102–9.
14. Lu J, Yang J, Batra D, Parik D. Hierarchical question-image co-attention for visual question answering. In: Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS); 2016 Dec 5–10; Barcelona, Spain. p. 289–97.
15. Ma C, Shen C, Dick A, Wu Q, Wang P, van den Hengel A, et al. Visual question answering with memory-augmented networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2018 Jun 18-23; 2018 Jun 18–23; Salt Lake City, UT, USA. p. 6975–84. doi:10.1109/CVPR.2018.00729.
16. Shi Y, Furlanello T, Zha S, Anandkumar A. Question type guided attention in visual question answering. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018 Sep 8–14; Munich, Germany. p. 158–75.
17. Gupta D, Suman S, Ekbal A. Hierarchical deep multi-modal network for medical visual question answering. Expert Syst Appl. 2021;164:113993. doi:10.1016/j.eswa.2020.113993.

18. Gao H, Mao J, Zhou J, Huang Z, Wang L, Xu W. Are you talking to a machine? Dataset and methods for multilingual image question answering. In: Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS); 2015 Dec 7–12; Montreal, QC, Canada. p. 2296–304.
19. Xu H, Saenko K. Ask, attend and answer: exploring question-guided spatial attention for visual question answering. In: Proceedings of the European Conference on Computer Vision (ECCV); 2016 Oct 11–14; Amsterdam, The Netherlands. p. 451–66.
20. Chen K, Wu X. VTQA: visual text question answering via entity alignment and cross-media reasoning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–22; Seattle, WA, USA. p. 27208–17. doi:10.1109/CVPR52733.2024.02570.
21. Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Zitnick CL, et al. VQA: visual question answering. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2015 Dec 7–13; Santiago, Chile. p. 2425–33. doi:10.1109/ICCV.2015.279.
22. Zhu Y, Groth O, Bernstein M, Li FF. Visual7W: grounded question answering in images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV, USA. p. 4995–5004. doi:10.1109/CVPR.2016.540.
23. Anderson P, He X, Buehler C, Teney D, Johnson M, Gould S, et al. Bottom-up and top-down attention for image captioning and VQA. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2018 Jun 18–23; Salt Lake City, UT, USA. p. 6077–86. doi:10.1109/CVPR.2018.00636.
24. Gao P, Jiang Z, You H, Lu P, Hoi SCH, Wang X, et al. Dynamic fusion with intra- and inter-modality attention flow for visual question answering. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019 Jun 15–20; Long Beach, CA, USA. p. 6632–41. doi:10.1109/cvpr.2019.00680.
25. Kamel SM, Hassan SI, Elrefaei L. VAQA: visual arabic question answering. Arab J Sci Eng. 2023;48(8):10803–23. doi:10.1007/s13369-023-07687-y.
26. Alsaleh SO, Bazi Y, Al Rahhal MM, Al Zuair M. Open-ended visual question answering model for remote sensing images. In: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS); 2022 Jul 17–22; Kuala Lumpur, Malaysia. p. 2848–51. doi:10.1109/IGARSS46834.2022.9884295.
27. Bazi Y, Rahhal MMA, Bashmal L, Zuair M. Vision-language model for visual question answering in medical imagery. Bioengineering. 2023;10(3):380. doi:10.3390/bioengineering10030380.
28. Liu G, He J, Li P, Zhong S, Li H, He G. Unified transformer with cross-modal mixture experts for remote-sensing visual question answering. Remote Sens. 2023;15(19):4682. doi:10.3390/rs15194682.
29. Yang Z, Garcia N, Chu C, Otani M, Nakashima Y, Takemura H. A comparative study of language transformers for video question answering. Neurocomputing. 2021;445:121–33. doi:10.1016/j.neucom.2021.02.092.
30. Huang X, Gong H. A dual-attention learning network with word and sentence embedding for medical visual question answering. IEEE Trans Med Imag. 2023;43(2):832–45. doi:10.1109/TMI.2023.3322868.
31. Yan F, Silamu W, Li Y. Deep modular bilinear attention network for visual question answering. Sensors. 2022;22(3):1045. doi:10.3390/s22031045.
32. Hackel L, Clasen KN, Ravanbakhsh M, Demir B. LIT-4-RSVQA: lightweight transformer-based visual question answering in remote sensing. In: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS); 2023 Jul 16–21; Pasadena, CA, USA. p. 2231–4. doi:10.1109/IGARSS52108.2023.10281674.
33. Abacha AB, Gayen S, Lau JJ, Rajaraman S, Demner-Fushman D. NLM at ImageCLEF 2018 Visual question answering in the medical domain. In: Proceedings of the Conference and Labs of the Evaluation Forum (CLEF); 2018 Sep 10–14; Avignon, France.
34. Osman A, Samek W. DRAU: dual recurrent attention units for visual question answering. Comput Vis Image Underst. 2019;185:24–30. doi:10.1016/j.cviu.2019.05.001.
35. Su Z, Zhu C, Dong Y, Cai D, Chen Y, Li J. Learning visual knowledge memory networks for visual question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2018 Jun 18–23; Salt Lake City, UT, USA. p. 7736–45. doi:10.1109/CVPR.2018.00807.
36. Lu S, Ding Y, Liu M, Yin Z, Yin L, Zheng W. Multiscale feature extraction and fusion of image and text in VQA. Int J Comput Intell Syst. 2023;16(1):54. doi:10.1007/s44196-023-00233-6.

37. Vu MH, Sznitman R, Nyholm T, Löfstedt T. Ensemble of streamlined bilinear visual question answering models for the ImageCLEF, 2019 challenge in the medical domain. In: Proceedings of the Conference and Labs of the Evaluation Forum; 2019 Sep 9–12; Lugano, Switzerland.
38. Sharma D, Purushotham S, Reddy CK. MedFuseNet: an attention-based multimodal deep learning model for visual question answering in the medical domain. Sci Rep. 2021;11(1):19826. doi:10.1038/s41598-021-98390-1.
39. Li Y, Yang Z, Hao T. TAM at VQA-Med 2021: a hybrid model with feature extraction and fusion for medical visual question answering. In: In; Proceedings of the Conference and Labs of the Evaluation Forum (CLEF); 2021 Sep 21–24; Bucharest, Romania.
40. Li Y, Long S, Yang Z, Weng H, Zeng K, Huang Z, et al. A bi-level representation learning model for medical visual question answering. J Biomed Inform. 2022;134:104183. doi:10.1016/j.jbi.2022.104183.
41. Jung B, Gu L, Harada T. Bumjun_jung at VQA-med 2020: vqa model based on feature extraction and multi-modal feature fusion. In: Proceedings of the Conference and Labs of the Evaluation Forum (CLEF); 2020 Sep 22–25; Thessaloniki, Greece.
42. Li J, Liu S. Lijie at ImageCLEFmed VQA-med 2021: attention model-based efficient interaction between multi-modality. In: Proceedings of the Conference and Labs of the Evaluation Forum (CLEF); 2021 Sep 21–24; Bucharest, Romania.
43. Upadhyay S, Tripathy SS. BIT mesra at ImageCLEF 2023: fusion of blended image and text features for medical VQA. In: Proceedings of the Conference and Labs of the Evaluation Forum (CLEF); 2023 Sep 18–21; Thessaloniki, Greece.
44. Liu F, Peng Y, Rosen MP. An effective deep transfer learning and information fusion framework for medical visual question answering. In: Proceedings of the 10th International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF); 2019 Sep 9–12; Lugano, Switzerland.
45. Shuang K, Guo J, Wang Z. Comprehensive-perception dynamic reasoning for visual question answering. Pattern Recognit. 2022;131:108878. doi:10.1016/j.patcog.2022.108878.
46. Li Y, Duan N, Zhou B, Chu X, Ouyang W, Wang X, et al. Visual question generation as dual task of visual question answering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2018 Jun 18–23; Salt Lake City, UT, USA. p. 6116–24. doi:10.1109/CVPR.2018.00640.
47. Bozinis T, Passalis N, Tefas A. Improving visual question answering using active perception on static images. In: Proceedings of the 25th International Conference on Pattern Recognition (ICPR); 2021 Jan 10–15; Milan, Italy. p. 879–84. doi:10.1109/icpr48806.2021.9412885.
48. Mao K. Enhancing visual question answering through bi-modal feature fusion: performance analysis. In: Proceedings of the 2024 6th International Conference on Image Processing and Machine Vision (IPMV); 2024 Jan 12–14; Macau, China. p. 115–22. doi:10.1145/3645259.3645278.
49. Zheng W, Yan L, Wang F-Y, Gou C. Learning from the guidance: knowledge embedded meta-learning for medical visual question answering. In: Proceedings of the International Conference on Neural Information Processing; 2020 Nov 18–22; Bangkok, Thailand. p. 194–202.
50. Miao Y, He S, Cheng W, Li G, Tong M. Research on visual question answering based on dynamic memory network model of multiple attention mechanisms. Sci Rep. 2022;12(1):16758. doi:10.1038/s41598-022-21149-9.
51. Mohamud SAM, Jalali A, Lee M. Encoder-decoder cycle for visual question answering based on perception-action cycle. Pattern Recognit. 2023;144:109848. doi:10.1016/j.patcog.2023.109848.
52. Malinowski M, Rohrbach M, Fritz M. Ask your neurons: a neural-based approach to answering questions about images. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2015 Dec 7–13; Santiago, Chile. p. 1–9. doi:10.1109/ICCV.2015.9.
53. Ren M, Kiros R, Zemel R. Exploring models and data for image question answering. In: Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS); 2015 Dec 7–12; Montreal, QC, Canada. p. 2953–61.
54. Ma L, Lu Z, Li H. Learning to answer questions from image using convolutional neural network. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI); 2016 Feb 12–17; Phoenix, Arizona. p. 3567–73.

55. Khare Y, Bagal V, Mathew M, Devi A, Priyakumar UD, Jawahar CV. MMBERT: multimodal BERT pretraining for improved medical VQA. In: Proceedings of the IEEE 18th International Symposium on Biomedical Imaging (ISBI); 2021 Apr 13–16; Nice, France. p. 1033–6. doi:10.1109/isbi48211.2021.9434063.
56. Silva JD, Magalhães J, Tuia D, Martins B. Remote sensing visual question answering with a self-attention multimodal encoder. In: Proceedings of the 5th ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery; 2022 Nov 1; Seattle, WA, USA. p. 40–9. doi:10.1145/3557918.3565874.
57. Seenivasan L, Islam M, Krishna AK, Ren H. Surgical-VQA: visual question answering in surgical scenes using transformer. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI); 2022 Sep 18–22; Singapore. p. 33–43.
58. Siebert T, Clasen KN, Ravanbakhsh M, Demir B. Multi-modal fusion transformer for visual question answering in remote sensing. In: Proceedings of the Image and Signal Processing for Remote Sensing; 2022 Sep 5–8; Berlin, Germany.
59. Naseem U, Khushi M, Kim J. Vision-language transformer for interpretable pathology visual question answering. IEEE J Biomed Health Inform. 2023;27(4):1681–90. doi:10.1109/JBHI.2022.3163751.
60. Kodali V, Berleant D. Recent, rapid advancement in visual question answering: a review. In: Proceedings of the IEEE International Conference on Electro Information Technology (eIT); 2022 May 19–21; Mankato, MN, USA. p. 139–46. doi:10.1109/eIT53891.2022.9813988.
61. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV, USA. p. 770–8. doi:10.1109/CVPR.2016.90.
62. Deng J, Dong W, Socher R, Li LJ, Kai L, Li FF. ImageNet: a large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2009 Jun 20–25; Miami, FL, USA. p. 248–55. doi:10.1109/CVPR.2009.5206848.
63. Soliman AB, Eissa K, El-Beltagy SR. AraVec: a set of Arabic word embedding models for use in Arabic NLP. Procedia Comput Sci. 2017;117:256–65. doi:10.1016/j.procs.2017.10.117.
64. Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014 Oct 25–29; Doha, Qatar. p. 1724–34. doi:10.3115/v1/d14-1179.
65. Google. Google Colaboratory [Online]. [cited 2024 Jun 1]. Available from: https://colab.research.google.com/.
66. Goyal Y, Khot T, Summers-Stay D, Batra D, Parikh D. Making the V in VQA matter: elevating the role of image understanding in visual question answering. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017 Jul 21–26; Honolulu, HI, USA. p. 6325–34. doi:10.1109/CVPR.2017.670.
67. Johnson J, Hariharan B, van der Maaten L, Li FF, Zitnick CL, Girshick R. CLEVR: a diagnostic dataset for compositional language and elementary visual reasoning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017 Jul 21–26; Honolulu, HI, USA. p. 1988–97. doi:10.1109/CVPR.2017.215.
68. Liu X, Dong Z, Zhang P. Tackling data bias in MUSIC-AVQA: crafting a balanced dataset for unbiased question-answering. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV); 2024 Jan 3–8; Waikoloa, HI, USA. p. 4466–75. doi:10.1109/WACV57701.2024.00442.
69. Xia H, Lan R, Li H, Song S. ST-VQA: shrinkage transformer with accurate alignment for visual question answering. Appl Intell. 2023;53(18):20967–78. doi:10.1007/s10489-023-04564-x.
70. Lobry S, Demir B, Tuia D. RSVQA meets bigearthnet: a new, large-scale, visual question answering dataset for remote sensing. In: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS); 2021 Jul 11–16; Brussels, Belgium. p. 1218–21. doi:10.1109/igarss47720.2021.9553307.
71. Abacha AB, Hasan SA, Datla VV, Liu J, Demner-Fushman D, Muller H. VQA-Med: overview of the medical visual question answering task at ImageCLEF 2019. In: Proceedings of Conference and Labs of the Evaluation Forum (CLEF); 2019 Sep 9–12; Lugano, Switzerland.
72. Pfeiffer J, Geigle G, Kamath A, Steitz JM, Roth S, Vulić I, et al. xGQA: cross-lingual visual question answering. In: Findings of the Association for Computational Linguistics (ACL); 2022 May 22–27; Dublin, Ireland.

73. Changpinyo S, Xue L, Yarom M, Thapliyal A, Szpektor I, Amelot J, et al. MaXM: towards multilingual visual question answering. In: Findings of the Association for Computational Linguistics; 2023 Dec 6–10; Singapore.
74. Becattini F, Bongini P, Bulla L, Del Bimbo A, Marinucci L, Mongiovì M, et al. VISCOUNTH: a large-scale multilingual visual question answering dataset for cultural heritage. ACM Trans Multimedia Comput Commun Appl. 2023;19(6):1–20. doi:10.1145/3590773.
75. Luu-Thuy Nguyen N, Nguyen NH, Vo DTD, Tran KQ, Nguyen KV. Evjvqa challenge: multilingual visual question answering. J Comput Sci Cybern. 2023;39(3):237–58. doi:10.15625/1813-9663/18157.
76. Chandrasekar A, Shimpi A, Naik D. Indic visual question answering. In: Proceedings of the IEEE International Conference on Signal Processing and Communications (SPCOM); 2022 Jul 11–15; Bangalore, India.