Multimedia Tools and Applications
https://doi.org/10.1007/s11042-023-14333-0

Video question answering supported by a multi-task learning objective

Alex Falcon[1,2] · **Giuseppe Serra**[2] · **Oswald Lanz**[3]

Received: 26 January 2022 / Revised: 30 June 2022 / Accepted: 2 January 2023

Abstract

Video Question Answering (VideoQA) concerns the realization of models able to analyze a video, and produce a meaningful answer to visual content-related questions. To encode the given question, word embedding techniques are used to compute a representation of the tokens suitable for neural networks. Yet almost all the works in the literature use the same technique, although recent advancements in NLP brought better solutions. This lack of analysis is a major shortcoming. To address it, in this paper we present a twofold contribution about this inquiry and its relation with question encoding. First of all, we integrate four of the most popular word embedding techniques in three recent VideoQA architectures, and investigate how they influence the performance on two public datasets: EgoVQA and PororoQA. Thanks to the learning process, we show that embeddings carry question type-dependent characteristics. Secondly, to leverage this result, we propose a simple yet effective multi-task learning protocol which uses an auxiliary task defined on the question types. By using the proposed learning strategy, significant improvements are observed in most of the combinations of network architecture and embedding under analysis.

Keywords Video question answering · Word embedding techniques · Vision and language · Multi-task learning

✉ Alex Falcon
afalcon@fbk.eu; falcon.alex@spes.uniud.it

Giuseppe Serra
giuseppe.serra@uniud.it

Oswald Lanz
lanz@inf.unibz.it

[1] Technologies of Vision, Fondazione Bruno Kessler, Via Sommarive, 18, Povo, 38123, Trento, Italy

[2] AILab, University of Udine, Via delle Scienze, 206, Udine, 33100, Udine, Italy

[3] Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, Bolzano, 39100, Italy

1 Introduction

Video Question Answering (VideoQA) is a task that requires to analyze and jointly reason on both the given video data and a visual content-related question, to produce a meaningful and coherent answer to it [15]. By solving this task, a computational model could reach human-level capabilities when dealing with both complex video and textual data, since it would require learning to reason about the elements of interest in the video and their spatial and temporal interactions related to the given question. VideoQA represents thus a challenging task at the interface between Computer Vision and Natural Language Processing (NLP) [6, 8, 55].

Typically, a VideoQA architecture consists of a video encoder, a text encoder, a fusion module, and a decoder to produce the final answer [7], as can be seen in Fig. 1a. These components for VideoQA are often built from neural networks which are the outcome of research work both from the NLP and Computer Vision communities. Deep convolutional networks originally proposed for Computer Vision tasks, such as image classification or action recognition, are usually employed as the backbone of the video encoder: as an example, among the many proposed architectures, appearance features are computed by means of VGG [41] in [5, 6], while [7, 13, 15, 16] adopt ResNet [11]; on the other hand, motion features can be produced by using C3D [46] (e.g. in [5, 6, 15, 16]) or BN-Inception [14], as in [7]. Similarly, text encoding involves the usage of word embedding techniques, which are algorithms that transform natural language words into fixed-size representations. Considering that these representations are suitable for neural network training, these techniques are responsible for the great developments in the NLP community in recent years, e.g. [31, 40] for the task of named entity recognition, and [3, 29] for text question-answering. Although several word embedding techniques with different characteristics have been proposed in the

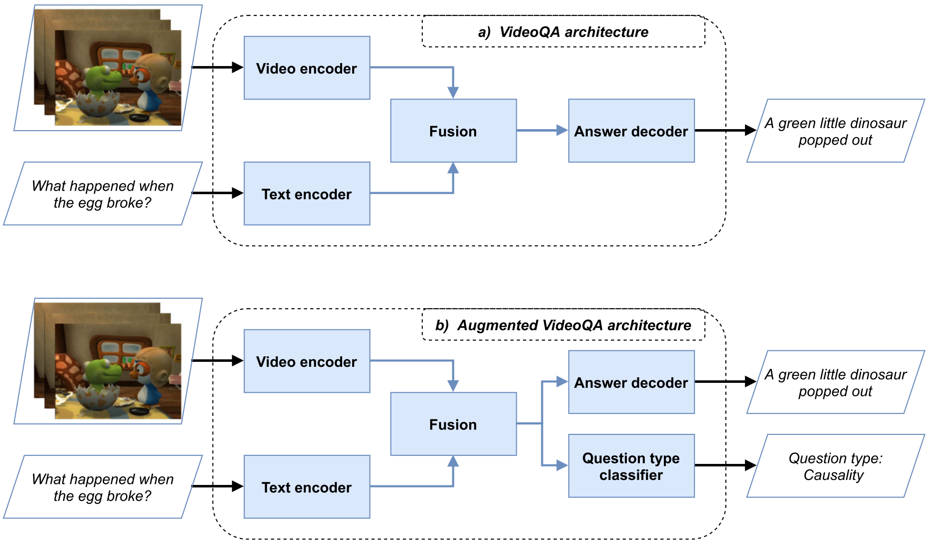

a) VideoQA architecture
Video encoder
Fusion
Answer decoder
Text encoder
What happened when the egg broke?
A green little dinosaur popped out
b) Augmented VideoQA architecture
Video encoder
Fusion
Answer decoder
Question type classifier
Text encoder
What happened when the egg broke?
A green little dinosaur popped out
Question type: Causality

Fig. 1 High level representation of a typical VideoQA architecture (shown in the upper part), consisting of: Video encoder and Text encoder which transform the raw input data into fixed-size representations; a Fusion module which combines the multimodal information; and the Answer decoder, which computes the final answer. In the lower part, we present an augmented VideoQA architecture which leverages a multi-task learning strategy in order to jointly classify and answer the input question

literature, VideoQA architectures rely on only a few of these techniques, such as GloVe proposed by [35] and word2vec by [33, 44]. As a consequence this language component, which provides the basis for the training process, is often underexplored in VideoQA architectures. Moreover, to the best of our knowledge, there are no complete and in-depth studies about the interaction between word embedding techniques and the VideoQA task. Hence, in this paper we propose an in-depth and extensive analysis to address these shortcomings.

Moreover, a multi-task learning strategy for VideoQA is introduced. As explained in a recent survey by [60], multi-task learning is a learning paradigm which aims at jointly learning multiple related tasks – in this way, the model needs to extract representations which are useful for all the considered tasks, therefore possibly leading to better generalization. This approach led to considerable improvements when applied to NLP (e.g. [37, 49]) and computer vision (e.g. [43, 59]), but also at the intersection of the two domains, especially when dealing with large scale visual-textual pretraining (e.g. [30, 56]). Few works in the literature introduce auxiliary tasks designed for VideoQA, such as the one by [18], where the model is trained to perform question answering, as well as video-subtitle alignment and temporal localization. In our work, an auxiliary task is introduced and it is designed with reference to the insights gathered from the aforementioned analysis of the word embedding techniques in the VideoQA domain.

In this paper, we propose a twofold contribution to VideoQA: firstly, a detailed analysis of word embedding techniques and of the final performance achieved by the model; secondly, a novel multi-task learning strategy to train a VideoQA architecture which aims at improving its generalization capabilities. In particular, we consider four word embedding techniques: GloVe, a popular technique which leverages co-occurrence statistics to compute low-dimensional embeddings; ELMo [36], a technique which uses character-level convolutions and LSTM networks; BERT [3] and XLM [24] which leverage Transformers [48] as part of their encoding process. We integrate and evaluate these four techniques into three different VideoQA architectures, each of which adopts multiple state-of-the-art techniques. As the main and most relevant result of our analysis, we observe that different word embedding techniques perform differently when facing specific question types. With the term 'question type' we refer to a categorization of the questions based on the target of the question itself. As an example, the question type 'Causality' refers to questions that ask to identify an event which happens in relation to another one. As can be seen in Fig. 1a, a question of this type could involve a specific event such as "what happened when the egg broke?", to which the model may correctly answer by pointing out what happened next, e.g. "a green little dinosaur popped out". As detailed in the experimental analysis, we observe that BERT and XLM exhibit a higher accuracy (with a sensible margin) than ELMo and GloVe when dealing with 'Causality' questions. To investigate the relation occurring between word embedding techniques and question types, we propose a solution involving multi-task learning (Fig. 1b) which, differently from traditional approaches to VideoQA, jointly optimizes both the task-oriented loss and a novel classification loss related to the question types.

The main contributions of this paper can be summarized as follows:

- we integrate four of the most adopted word embedding techniques (GloVe, ELMo, BERT, and XLM) in three recent VideoQA architectures, from an attention-based encoder-decoder baseline [15] to more complex architectures involving memory [8] and reasoning [6];
- by quantitatively analyzing all the 12 combinations of embedding techniques and VideoQA architectures, we observe that word embedding techniques work better for specific question types;

- we propose a simple yet effective multi-task learning strategy which can help the considered models achieve better generalization, leading to considerable improvements on two public datasets;
- we release code and pretrained models at https://github.com/aranciokov/MT-VideoQA to support research in this important field, to ease the reproducibility of the results, and to provide a codebase adaptable to different VideoQA datasets and models.

The rest of the paper is organized as follows. In Section 2 a comprehensive literature review is performed in order to contextualize the proposed method. The proposed methodology is presented in detail in Section 3. Several experimental results are shown and discussed in Section 4, concerning both the analysis of multiple word embedding techniques and the proposed multi-task learning strategy. Finally, Section 5 concludes the paper.

2 Related work

In this section we discuss the work related to the two main topics involved in our study, i.e. Video Question Answering, and word embedding techniques.

2.1 Video question answering

Thanks to the recent availability of several large scale VideoQA datasets, such as TVQA [25], How2VQA69M [58], TGIF-QA [15], MSRVTT-QA and MSVD-QA [55], this task has gained more and more attention by researchers in both Computer Vision and NLP fields [5, 6, 8, 13, 15, 34, 55, 57, 58]. In particular, two types of tasks are often linked to VideoQA: the "open-ended" (e.g. in [5, 15, 55, 58]) and the "multiple choice" task (e.g. [5, 15, 19, 25, 45]). The former is usually treated as a classification problem where the correct answer is identified in a predefined set of possible answers, although it can also be approached through generative techniques by generating a free-form response word-by-word, e.g. in [54, 61]; the latter (i.e. "multiple choice", which is also the task that we tackle within this work) involves the usage of a small pool of candidate answers (e.g. five choices in [5, 15]) which are possibly different for every question, and the model selects one of the candidates based on a score computed through a regressor. Considering that in our paper we describe and apply our approach in relation to the multiple choice task, in the following we focus on this specific task.

A prominent research direction for the multiple choice task consists in the usage of deep neural networks to learn suitable temporal or spatio-temporal features, eventually adopting attention mechanisms to filter out irrelevant features or redundant frames or frame regions, e.g. [8, 15, 25, 28]. "ST-VQA" by [15] integrated a temporal attention module to attend to the most important frames in the input clip, while leveraging LSTM networks to model the sequential aspect of both the visual and textual data. Conversely, [28] proposed a Positional Self-Attention block based on [48] to replace recurrent networks, while also using self-attention to learn self-attended single-modality features and a cross-modal attention mechanism in order to compute rich representations for the available visual and textual data. Although these methods led to considerable improvements on several public benchmarks, an important drawback is that they relied on clip- or frame-level representations, therefore missing out finer-grained details at the object-level. A recent research direction which focused on this aspect explored local relations between the visible objects and their natural language description. Huang et al. [13] built a complete graph using frame- and object-level features as node descriptors, making the graph location-aware by augmenting the nodes by means of spatial and temporal position features, and then reasoned over this structure with a Graph Convolutional Network [22]. Yet, only visual information are used to build and reason on the graph structure. Jiang and Han [16] argued that visual and linguistic factors have coordinated semantics which can be aligned to perform cross-modal reasoning, hence leading to the construction of an heterogeneous data structure. Although multiple video modalities are used by Jiang and Han, the semantic relations between them are not fully used. Therefore, [34] suggested to model both the visual-linguistic interactions as well as the semantic relations between different video modalities (e.g. appearance, motion) by using the question as a proxy. Differently from these works, a new research direction shifted the attention to the training strategy. In particular, objective functions taken from the NLP domain were adapted to the VideoQA task. Yang et al. [57] performed masked language modeling (MLM) and next sentence prediction using object-level and question features as one of the inputs, while the candidate answers are used as possible next sentences. Similarly, [58] suggested using MLM and a contrastive objective in order to choose the correct answer using similarity metrics. Several papers (e.g. [26, 27, 62]) have also achieved notable performance on the target dataset by performing a large scale pretraining phase on large scale multi-modal datasets such as VisualGenome [23], HowTo100M [32], or How2VQA69M [58] by using language-only, vision-only, or language-vision proxy tasks. Yet, a major drawback of these pretraining procedures is the prohibitive computational cost, e.g. the training procedure on How2VQA69M lasted 2 days while using 8 Tesla V100 GPUs, according to [58]. Finally, given the sequential nature of the data involved in VideoQA, the usage of memory layers has also been explored, raising the possibility to interact with a memory made of multiple vectors, which is typically not possible in other neural networks which have a memory consisting of a single vector. "CoMem" [8] used memory layers to generate attention cues starting from both motion and appearance features. "HME-VQA" [6] introduced an heterogeneous memory layer while also proposing a multi-step LSTM-based reasoning technique. In our work, among all the aforementioned solutions, we chose to use ST-VQA, CoMem, and HME-VQA because they offer increasingly complex and rich solutions which cover multiple state-of-the-art techniques, while also offering open source code bases. In particular, ST-VQA offers an attention-based encoder-decoder, CoMem also employs memory layers to support the generation of attention cues from both video modalities, and finally HME-VQA integrates multi-step reasoning as well. Although these works use advanced techniques to perform the video modeling or to fuse heterogeneous types of information, they only explore one technique to embed the words into vectorial representations, that is GloVe, therefore ignoring recent advancements in NLP. To this end, given that understanding the question is fundamental to predict the correct answer, in this paper we analyze how four popular embedding techniques interact with the network architectures used for VideoQA. Then, we propose a multi-task learning strategy to improve the generalization capabilities of a VideoQA system, by designing an auxiliary task based on the results of the preliminary analysis.

2.2 Word embedding techniques

NLP has rapidly evolved during the past few years and one of the most investigated topics is related to neural language models (LM). Before the introduction of BERT, GloVe and ELMo were two of the most used techniques.

Pennington et al. [35] introduced GloVe, which is a *static*, non-contextual word embedding technique which computes the word vectors by leveraging both local and global statistics (e.g. word co-occurrence), assigning the same embedding (i.e. a real-valued vector) to a word independently from the context in which such word is used. Differently from GloVe, [36] proposed a contextual (i.e. each word receives an embedding depending on the context) LSTM-based LM called ELMo. Moreover, the objective of ELMo is to estimate the probability distribution of the training corpus using recurrence, and is thus classified as an autoregressive LM.

Lately, the introduction of BERT by [3] and its variants (e.g., XLM by [24] and DistilBERT by [38]) showed that these models have strong transfer learning capabilities by simply attaching and training a task-specific head over the pretrained backbone. Being based on Transformers [48], they are solely based on attention mechanisms and do not use any recurrent neural network. Because of this they are classified as autoencoding LMs: instead of estimating the probability distribution of the corpus, they learn a function to reconstruct the input from masked versions of it.

With the introduction of BERT, several tasks in NLP reached new state-of-the-art results, yet its predecessors are still used in many works, e.g. GloVe in [6, 13, 16, 34, 50]. As mentioned before, to fairly analyze the influence of the most important word embedding techniques in the VideoQA task, we propose a study to understand which one to use by integrating each of them in three VideoQA models.

3 Methodology

In our study we explore the usage and integration of several word embedding techniques into three different VideoQA architectures (i.e. ST-VQA, CoMem, and HME-VQA) which involve multiple state-of-the-art techniques including attention mechanisms, memory layers, and multi-step reasoning. To treat them all in a shared but comprehensive manner, we present in Fig. 2 a detailed overview of a VideoQA architecture: it comprises both the common components, such as "Feature extraction" and "Word embedding", as well as the modules which are exclusive to only some of the architectures, such as the "Reasoning module" which is only used by HME-VQA. Moreover, on the right we also outline the additional

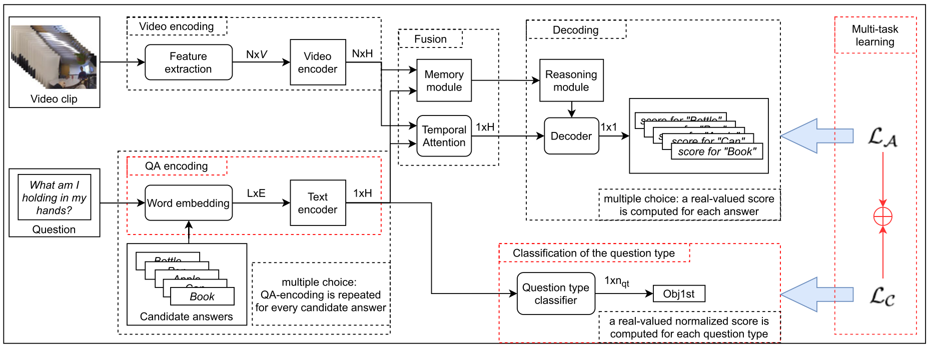


Fig. 2 General architecture of the models considered in our study, which focuses on the Word Embedding module and the Question type classifier (outlined in red). The former receives the question and the candidate answers, and outputs L embeddings of size E. The latter is trained in a multi-task learning style and we show it helps improving the performance

components related to the proposed multi-task learning strategy, including a module used to classify the question type, and the joint loss function (built on $\mathcal{L}_{\mathcal{A}}$ and $\mathcal{L}_{\mathcal{C}}$, described in Section 3.1). As already mentioned, a VideoQA architecture can be seen as made of four blocks, that is Video encoding, QA encoding, Fusion, and Decoding. Given the input data, we extract a sequence of embeddings for both the video (in "Feature extraction") and the input question (in "Word embedding"), as well as for the candidate answers. For the video, VGG is employed to extract appearance features, while C3D is used for motion features. For the textual data, we use one of the word embedding techniques that we explore in this paper (see Section 4.2 for more details). These two steps are done for all the architectures that we consider, which are ST-VQA [15], CoMem [8], and HME-VQA [6]. Then, for both visual and textual data, we employ an encoder made of two stacked LSTM networks to model the evolution of the features. Note that ST-VQA concatenates appearance and motion features before processing them by using the Video encoder; CoMem and HME-VQA independently model the two sequences of features via two independent Video encoders which follow the same structure. The Fusion block aims at computing a representation which takes both the video and textual information into account. In ST-VQA, this is done through a Temporal Attention module, which weighs each visual features vector based on the aggregated textual representation; in CoMem, appearance and motion features are used to provide attention cues to each other by employing a Memory module; finally, a Reasoning module is used in HME-VQA to compute an aggregated representation of the output of the heterogeneous memory layer, while also employing two temporal attention modules to compute modality-independent attention-weighted vectors (see Section 3.2 for an in-depth explanation). The fused features are then used in the decisional process to predict a regression score for the candidate answer. Note that in the case of HME-VQA the input to the decoder uses both the attended visual vectors and the output of the Reasoning module. To optimize the network parameters, an hinge loss is used to enforce a margin (e.g. 1, as in (2)) between the score computed for the correct answer and all the other candidate answers.

In Section 3.1 we thoroughly describe the proposed multi-task learning strategy. For completeness, we also provide further details about the adopted methods in Section 3.2, by focusing on their differences.

3.1 Multi-task learning strategy

When asking a question to a VideoQA model, we expect it to extract visual and textual information which are related to the question itself. Furthermore, we expect questions of the same category to share a similar visual and textual joint representation as computed by the Fusion module. As an example, questions asking to identify an object may require spatial features which are closely related to the objects shown in the video, while asking to recognize an action may shift the focus on temporal aspects. For this reason, we propose to incorporate the question type (as a classification objective) into the loss function we strive to optimize.

The proposed multi-task learning strategy involves a joint loss function, comprising of a pairwise hinge loss $\mathcal{L}_{\mathcal{A}}$, which is used to train the model for the VideoQA multiple choice task, as it is often done in the literature (e.g. in [5, 15]) and a classification loss $\mathcal{L}_{\mathcal{C}}$ which we use to make the model able to categorize an input question into one of the predetermined types. Such a joint loss can be described as:

$$\mathcal{L} = \mathcal{L}_{\mathcal{A}} + \mathcal{L}_{\mathcal{C}} \tag{1}$$

For a given input sample, the pairwise hinge loss can be described as:

$$\mathcal{L}_{c,r} = \begin{cases} 0 & \text{if } c = r \\ \max(0, 1 + s_c - s_r) & \text{if } c \neq r \end{cases} \tag{2}$$

where s_c and s_r are the scores d_r computed by the Decoder (see Section 3.2, (11) for more details) for the candidate answer c and the right answer r. To compute $\mathcal{L}_{c,r}$ for each sample in the minibatch, we use the following equation:

$$\mathcal{L}_{\mathcal{A}} = \sum_{q \in \mathcal{Q}} \sum_{c \in \mathcal{C}_q} \mathcal{L}_{c,r} \tag{3}$$

where $\mathcal{Q}$ represents the questions in the minibatch, while $\mathcal{C}_q$ and r are respectively the set of candidate answers and the correct answer for q.

To deal with the additional classification objective $\mathcal{L}_{\mathcal{C}}$, we augment all the considered architectures by attaching a classifier head on top of the Text encoder:

$$l_{qt} = softmax(\epsilon_w W_{qt} + b_{qt}) \tag{4}$$

where $\epsilon_w \in \mathbb{R}^{1 \times H}$ is the output of the Text encoder (see Section 3.2 for more details), $W_{qt} \in \mathbb{R}^{H \times n_{qt}}$ and $b_{qt} \in \mathbb{R}^{1 \times n_{qt}}$ are trainable parameters, H is twice the hidden size h, and finally n_{qt} is the amount of question types in the considered dataset. To train the model for this additional task, we consider the following equations:

$$\chi(x, y) = \frac{1}{n_{qt}} \sum_{i=1}^{n_{qt}} -(y_i \cdot \log(x_i) + (1 - y_i) \cdot \log(1 - x_i)) \tag{5}$$

$$\mathcal{L}_{\mathcal{C}} = \frac{1}{|\mathcal{Q}| \cdot |\mathcal{C}_q|} \sum_{q \in \mathcal{Q}} \sum_{c \in \mathcal{C}_q} \chi(l_{qt}, \text{one-hot}(t)) \tag{6}$$

where t is the type of the question q, and one-hot(t) computes its one-hot representation. By using l_{qt} we consider the question as well as the candidate answer because both may contain helpful and discriminative information while optimizing for this task.

As previously mentioned, we apply our multi-task learning strategy to several different architectures, in order to show its general applicability. In the following section, we provide a more detailed presentation of the considered VideoQA architectures.

3.2 VideoQA architectures

Here we describe three VideoQA models which can be seen as made of four blocks [7]: Question-Answer (QA) Encoding, Video Encoding, Fusion, and Decoding. This can be observed both in Fig. 1, where we depict it from a high level view, and in Fig. 2, which shows a general framework to cover all the models used in this study. These three models involve several state-of-the-art techniques, including attention mechanisms, memory layers, and multi-step reasoning, offering an heterogeneous experimental setting. In Fig. 3 we also include a more in-depth view on the three architectures in order to highlight the major differences between them, which are also commented in the following subsections. The four blocks previously identified in Fig. 2 are respectively colored in purple, blue, yellow, and darker yellow. The proposed multi-task learning strategy (see Section 3.1) is highlighted in red. As can be seen, the auxiliary task introduced in this paper, that is the prediction of the question type, is performed by using the textual features computed by the LSTM-based

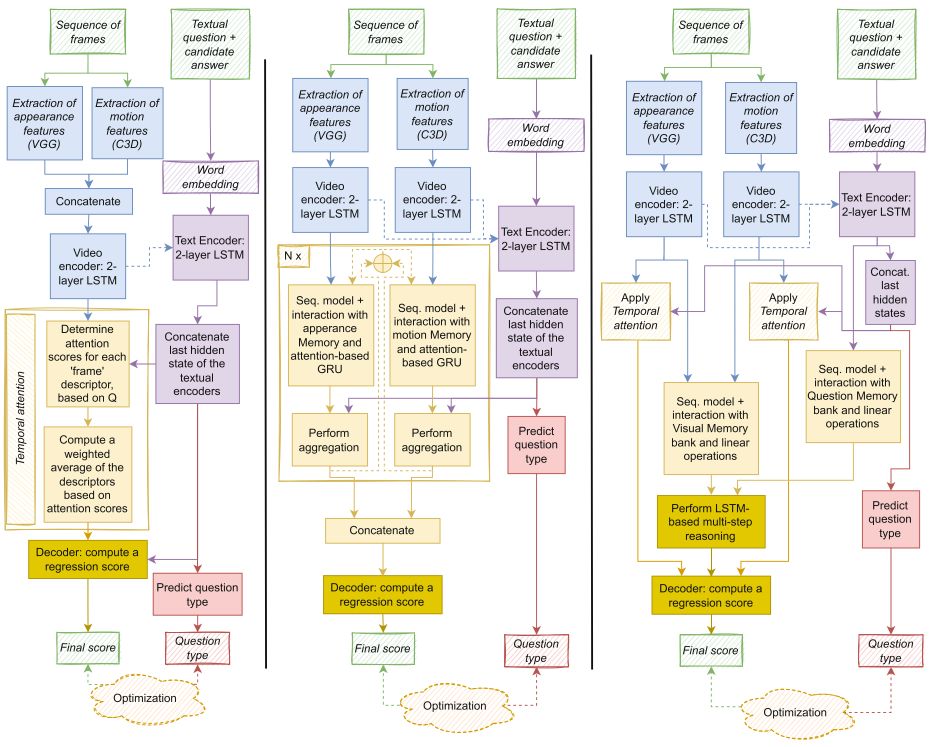

Sequence of frames
Textual question + candidate answer
Extraction of appearance features (VGG)
Extraction of motion features (C3D)
Word embedding
Concatenate
Video encoder: 2-layer LSTM
Text Encoder: 2-layer LSTM
Temporal attention
Determine attention scores for each 'frame' descriptor, based on Q
Compute a weighted average of the descriptors based on attention scores
Concatenate last hidden state of the textual encoders
Decoder: compute a regression score
Predict question type
Final score
Question type
Optimization
N x
Seq. model + interaction with apperance Memory and attention-based GRU
Seq. model + interaction with motion Memory and attention-based GRU
Perform aggregation
Apply Temporal attention
Concat. last hidden states
Seq. model + interaction with Visual Memory bank and linear operations
Seq. model + interaction with Question Memory bank and linear operations
Perform LSTM-based multi-step reasoning

Fig. 3 Detailed diagram of the three models we selected from the literature, *i.e.* (**left**) ST-VQA by [15], (**middle**) CoMem by [8], and (**right**) HME-VQA by [6]. Compared to Fig. 2, we color in blue the "Video encoding", in purple the "QA encoding", in yellow the "Fusion", in darker yellow the "Decoding", and finally we highlight in red the modification applied to the base algorithms in order to use the proposed multi-task learning strategy. In the "Optimization" cloud we perform $\mathcal{L}_{\mathcal{A}} + \mathcal{L}_{\mathcal{C}}$

Text Encoder. The proposed multi-task learning strategy is easily extendable to heterogeneous architectures and, in fact, in Section 3.1 it is shown how to apply it to three different techniques from the literature.

ST-VQA The first model we use is based on ST-VQA proposed by [15], an encoder-decoder architecture supported by attention mechanisms. Since we deal with the multiple choice task, the QA encoding module receives a question and a pool of candidate answers. Let $q_1 \dots q_m$ and $a_1 \dots a_n$ be the sequence of m tokens of the question and n tokens of (one of the candidate) answer. As shown in Fig. 2, the encoding of question and candidate answer is performed for each of the candidates, since they are (possibly) different for each question. In Fig. 3 (left) this is shown as the "Textual question + candidate answer" block. To do so, $q_1 \dots q_m$ and $a_1 \dots a_n$ are concatenated into δ and used as input to the embedding technique (shown as "Word embedding" in Fig. 3), eventually adding some special tokens (for more details, see Section 4.2). Hence, the textual data are first embedded into $\phi_w \in \mathbb{R}^{L\times E}$, where L is the number of tokens in question and answer, and E is the embedding size. Then, ϕ_w is input to the Text Encoder, which consists of two stacked LSTM networks. The encoded textual features ϵ_w are obtained by concatenating the *last* hidden state of both the LSTM networks, thus forming a feature vector $\epsilon_w \in \mathbb{R}^{1\times H}$. In the Video Encoding block, both motion and appearance features are obtained from an input video clip made of N frames. Both the feature extraction and the Video Encoder are depicted in Fig. 3 with the blue color. To compute the appearance features, they use a frozen VGG-16, pretrained on ImageNet, and extract the *fc7* activations ($\phi_a \in \mathbb{R}^{N\times 4096}$). To compute the motion features, they use a frozen C3D, pretrained on Sports1M [17] and fine-tuned on UCF101 [42], and extract the *fc7* activations ($\phi_m \in \mathbb{R}^{N\times 4096}$). In our work, we use VGG-16 and C3D because the feature vectors are computed through a transformation of the feature maps computed by the convolutional layers, and not by employing a global pooling layer. In fact, while the usage of the latter operation (for example, employed in ResNet by [11]) greatly reduces the quantity of parameters in the model, it also leads to a loss of the positional information available in the activation tensors. The features extracted from VGG and C3D are concatenated obtaining $\phi_{a,m} \in \mathbb{R}^{N\times V}$ (with $V = 8192$) and then input to a Video Encoder made of two stacked LSTM networks. Despite similar in structure to the Text Encoder, the output of the Video Encoder consists of the full sequence of hidden states, i.e. $\epsilon_v \in \mathbb{R}^{N\times H}$.

ST-VQA features an attention-based [1, 12] Fusion block, shown in yellow in Fig. 3 (left), which lets the model learn which frames are more important based on the encoded textual features. It receives in input the encoded video features ϵ_v and the textual features ϵ_w, and can be described by the following equations:

$$\omega_s = tanh(\epsilon_v W_v + \epsilon_w W_w + b_s) W_s \tag{7}$$

$$\alpha_s = softmax(\omega_s) \tag{8}$$

$$\omega_a = \mathbb{1} \cdot (\alpha_s \circ \epsilon_v) \tag{9}$$

where $W_v, W_w \in \mathbb{R}^{H\times h}$, $W_s \in \mathbb{R}^h$, and $b_s \in \mathbb{R}^{1\times h}$ are learnable parameters. Equation (9) implements a sum-pool operation, where $\mathbb{1}$ is a row of ones ($1^{1\times N}$). $\circ$ represents the element-wise multiplication operator.

Finally, the decoder we use is based on the one proposed in [5]. Decoding is done for each QA pair, that is in our multiple choice setting, five times with different textual features producing five different scores, one per candidate answer. It can be described by the following equations:

$$d_f = tanh(\omega_a W_a + b_a) \tag{10}$$

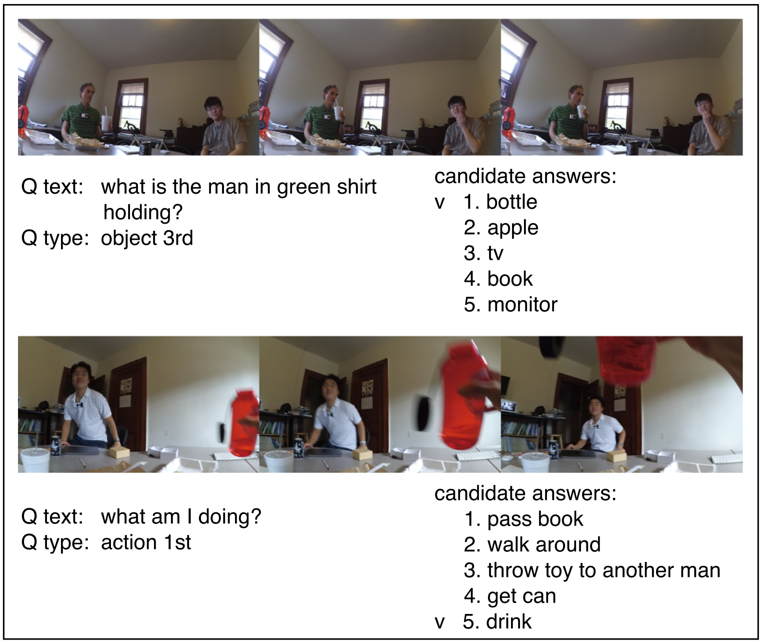
$$d_r = (d_f \circ \epsilon_w) W_d + b_d \tag{11}$$

where $W_a \in \mathbb{R}^{H\times H}$, $W_d \in \mathbb{R}^{H\times 1}$, and $b_a \in \mathbb{R}^{1\times H}$ are parameters, $d_f \in \mathbb{R}^{1\times H}$, $d_r \in \mathbb{R}$. d_r is the score obtained by testing a specific candidate answer (out of the five possible choices related to the given question). The Decoding step is shown with a darker shade of yellow in Fig. 3.

CoMem The CoMem model is based on the work by [8]. As in ST-VQA the textual features are computed by the word embedding technique and the Text Encoder. The visual features are again extracted using VGG and C3D but, in this case, they are not concatenated and they are encoded with two independent Video Encoders. Furthermore, hidden and cell state of the Text Encoder are initialized with those of the Video Encoder. Yet, the main difference with the ST-VQA approach is the usage of a Memory module within the Fusion block, shown in yellow in Fig. 3 (middle), which is supported by a co-attention mechanism. That is, they show appearance features are useful to guide the extraction of relevant motion features, and vice versa. To capture these interactions, both attention and memories are exploited. Moreover, the Memory module is used sequentially in the architecture as a fusion technique by replacing the Temporal Attention. These operations are shown in Fig. 3 (middle) in yellow.

In particular, CoMem uses and iteratively updates two memories called “appearance memory” and “motion memory”: at every iteration, both are updated by an attention function which jointly attends to both motion and appearance encoded features, the memory, and the question embedding. Then, appearance and motion features are used to update each memory (shown with the $\bigoplus$ operator in Fig. 3). This operation is repeated a fixed amount of times and is depicted with the “N x” block.

HME-VQA As in CoMem, HME-VQA [6] follows a similar overall flow: the visual features computed by using VGG and C3D with independent Video Encoders, and the textual features are computed with the Text Encoder applied on top of the word embeddings. Differently from CoMem, HME-VQA uses two memories, a “visual memory” and a “question memory”, as depicted in Fig. 3 (right). The former is updated by an attention function that exploits three hidden states, which consider appearance and motion features both separately and jointly. In the latter only one hidden state is used. A second novelty in HME-VQA is the usage of an LSTM-based Reasoning module (colored with a dark yellow in Fig. 3), which consists of three steps: first of all two context vectors, c_v and c_q, are created by attending to the hidden states of the “visual memory” and the “question memory”, and the previous hidden state s_{t-1}; then c_v, c_q, and s_{t-1} are used to separately compute attention weights, which are used to compute the “fused knowledge”, i.e. a weighted sum of c_v and c_q; finally, the LSTM updates s_t using the fused knowledge and s_{t-1}. The last hidden state of the LSTM is used as a distilled version of the given data. A Temporal attention module is separately applied on the appearance and motion features, as shown in Fig. 3. Finally, the text-attended



Q text: what is the man in green shirt holding?
Q type: object 3rd

candidate answers:
v 1. bottle
2. apple
3. tv
4. book
5. monitor

Q text: what am I doing?
Q type: action 1st

candidate answers:
1. pass book
2. walk around
3. throw toy to another man
4. get can
v 5. drink

Fig. 4 Samples of clips, questions, and candidate answers from EgoVQA

visual features and the output of the Reasoning module are used in conjunction with the Decoder to compute the score for the answer.

4 Results and discussions

To perform the analysis of the word embedding techniques and to validate our multi-task learning strategy, we choose to use two public VideoQA datasets, PororoQA [19] and EgoVQA [5], as they also briefly discuss question types. After presenting these datasets, we thoroughly describe the word embedding techniques that we used, and we discuss both the overall results and the per question type results.

4.1 The datasets

EgoVQA Presented in [5], it features more than 600 QA pairs and the same number of clips, which are 20-100 seconds long and are obtained from 16 egocentric videos (5-10 minutes long) based on 8 different scenarios. An example of these egocentric videos and QA pairs can be seen in Fig. 4. The questions can be grouped in eight types, as described in Table 1.

For each video and question five candidate answers are provided, of which only one is correct. The wrong answers are randomly sampled from a candidate pool based on the question type, i.e. if the question requires to recognize an action, the five candidates (the right one and the four wrong) are actions.

PororoQA Introduced by [19], it features around 8,800 QA pairs over 6,160 clips, which are 3.5 seconds long (on average) and are obtained from 166 episodes of the Korean cartoon “Pororo”. PororoQA follows the multiple choice setting with five candidates, and the question types are shown in Table 2. Although this dataset also offers scene descriptions and subtitles, we choose not to use them because it would be a different task (Video Story Question Answering, see [9, 25, 45] as well), and thus out of the scope of this paper. For this reason, we are only using RGB frames, hence why we are not comparing the performance results we obtain with [19, 20] and [57], where scene descriptions and subtitles are exploited as well.

Table 1 Description of the question types available in the EgoVQA dataset

Code	Description	Example
Act_{1st}	action performed by camera wearer	“what am I doing”
Act_{3rd}	action performed by different actor	“what is the man in red clothes doing”
Obj_{1st}	object the camera wearer is interacting with	“what am I holding in my hands”
Obj_{3rd}	object a different actor is interacting with	“what is placed on the desk”
Who_{1st}	who the camera wearer is interacting with	“who am I talking with”
Who_{3rd}	who is performing a certain action	“who is eating salad”
Cnt	number of persons or objects in the scenes	“how many people am I talking with”
Col	identify the color of an object	“what is the color of the toy in my hands”

Table 2 Description of the question types available in the PororoQA dataset

Code	Description	Example
Abs	questions about abstract concepts	"what is the weather like in the forest"
Act	recognize the action performed	"what are Pororo and Crong doing"
$Caus$	describe which event follows another one	"what happened when the egg broke"
Det	detail of something in the clip	"what kind of bird is Pororo"
Loc	describe where an event takes place	"where did Pororo take the egg"
Met	describe *how* something is done	"how did Pororo introduce himself"
Per	identify who did a specific action	"who was sliding on the ice"
$Reas$	motivate a specific event	"why is Pororo running away"
$Stmt$	questions about the content of a speech	"what did the baby dinosaur say first"
$Time$	describe when an event takes place in the clip	"when does Pororo find an egg"
Y/N	yes/no questions with an explanation	"are Pororo's friends scared of the dinosaur"

4.2 Word embeddings

In our experiments, we choose to use GloVe, ELMo, BERT, and XLM because of their popularity and because they provide both contextual and non-contextual embeddings. Note that they use different tokenizers: in particular we use full words for GloVe and ELMo, WordPiece [53] for BERT, and Byte-Pair Encoding [39] for XLM.

GloVe By using GloVe, pretrained on the Common Crawl dataset , a vector of size $E = 300$ is computed for each word in both question and answer. Since GloVe is not contextual, question and answer can be given in input to it either jointly or separately obtaining the same embedding. In the former case, the input to GloVe is a simple concatenation of the tokens, i.e. $\delta = q_1 \dots q_m a_1 \dots a_n$, whereas the output is $\phi_w \in \mathbb{R}^{(n+m)\times E}$. In the latter case, two embedded representations are computed by separately using GloVe on $q_1 \dots q_m$ and $a_1 \dots a_n$, which are then concatenated to obtain ϕ_w.

ELMo ELMo is a contextual word embedding technique based on LSTMs which computes for each word multiple representations, derived from its hidden states. In our setting, we extract the topmost representation of size $E = 1024$. Since ELMo is contextual, as opposed to GloVe which is not, the word embeddings for question and answer need to be jointly computed, i.e. the input to ELMo is $\delta = q_1 \dots q_m a_1 \dots a_n$, with $|\delta| = L$.

BERT Similarly to ELMo, BERT computes multiple representations for each word. We use the base version consisting of 12 attention heads and 12 layers, each of which produces a

Table 3 Average accuracy over EgoVQA using the frozen embeddings

	GloVe	ELMo	BERT	XLM
ST-VQA	$35.7_{\pm 4.2}$	$34.9_{\pm 4.5}$	$36.1_{\pm 6.0}$	$24.0_{\pm 3.3}$
CoMem	$34.1_{\pm 3.8}$	$33.6_{\pm 4.8}$	$31.6_{\pm 5.3}$	$25.7_{\pm 3.9}$
HME-VQA	$35.4_{\pm 3.1}$	$35.5_{\pm 3.8}$	$33.6_{\pm 4.5}$	$26.8_{\pm 3.2}$

Table 4 Average accuracy over PororoQA using the frozen embeddings

	GloVe	ELMo	BERT	XLM
ST-VQA	$36.8_{\pm 0.5}$	$35.7_{\pm 0.9}$	$39.7_{\pm 0.9}$	$27.4_{\pm 1.2}$
CoMem	$33.3_{\pm 1.1}$	$35.4_{\pm 1.0}$	$36.1_{\pm 0.8}$	$22.2_{\pm 0.7}$
HME-VQA	$36.8_{\pm 1.8}$	$34.5_{\pm 1.1}$	$39.2_{\pm 0.7}$	$27.8_{\pm 0.9}$

different embedding of size $E = 768$. We use the embeddings from the last layer. For BERT, $\delta = \alpha q_1 \dots q_m \sigma a_1 \dots a_n \sigma$, where α is the token '[CLS]', and σ is the separator '[SEP]'.

Note that although BERT already provides an aggregated output in the representation of the '[CLS]' token, we chose to also adopt the LSTM-based Text Encoder (see Section 3.2) on top of it because of two reasons: firstly, to have an overall similar structure across all four embedding techniques; secondly, because it can provide further context while also improving the final performance [9].

XLM XLM is a variant of BERT which uses a different training technique and also uses BERT as an initialization step for machine translation models. In particular, we adopt the base version of XLM, which uses 12 layers and 16 attention heads. The word embeddings computed using this method have size $E = 2048$. δ is defined in the same way as for BERT.

4.3 Evaluation protocol

In our setting, we fix $H = 512$ and $h = 256$. To optimize the parameters we use Adam [21] with a fixed learning rate of 10^{-3} and a batch size of 8.

To implement our solution we use Python 3.6, Numpy 1.18, and PyTorch 1.7. We use AllenNLP [10] to test ELMo[1], and the 'transformers' library 3.5.1 [52] to test BERT[2] and XLM.[3]

To evaluate the performance of the multiple combinations explored in this paper, we train for 20 epochs, then we select the model with the best validation accuracy and use it for testing. This is done five times (fixed seeds, 0 to 4), in order to obtain more stable and reliable results. It is particularly important for EgoVQA, where the amount of available data is relatively low and thus susceptible to highly variable results over multiple runs.

4.4 Results using the frozen embeddings

The first set of results analyze how different embedding techniques affect the final performance, while using them pretrained and frozen. We start from this experiment because word embeddings are often kept frozen and not trained, e.g. in [51], since they are learned on big text corpora from which the embeddings gather semantics which can transfer well to downstream tasks. Tables 3 and 4 present the overall accuracy for EgoVQA and PororoQA, and show that BERT provides the best embeddings for the task: in particular, adopting BERT in the ST-VQA architecture leads to an average accuracy of 36.1% in EgoVQA (Table 3) and

[1]pretrained: 'elmo_2x4096_512_2048cnn_2xhighway_weights'

[2]pretrained: 'bert-base-uncased'

[3]pretrained: 'xlm-mlm-en-2048'

Table 5 Accuracy per question type over EgoVQA using the frozen embeddings

Met;Emb	Question type accuracy (%)							
	Act_{1st}	Act_{3rd}	Obj_{1st}	Obj_{3rd}	Who_{1st}	Who_{3rd}	Cnt	Col
S; G	$29.2_{\pm 2.2}$	$33.9_{\pm 2.2}$	$42.6_{\pm 4.1}$	$37.0_{\pm 4.9}$	$52.3_{\pm 3.1}$	$40.3_{\pm 5.4}$	$38.8_{\pm 2.5}$	$20.0_{\pm 4.3}$
S; E	$29.2_{\pm 1.8}$	$32.6_{\pm 3.4}$	$41.1_{\pm 4.6}$	$35.6_{\pm 2.2}$	$49.2_{\pm 3.8}$	$41.6_{\pm 2.1}$	$38.4_{\pm 3.6}$	$18.1_{\pm 4.4}$
S; B	$28.9_{\pm 2.8}$	$35.9_{\pm 1.8}$	$39.6_{\pm 3.2}$	$35.6_{\pm 2.8}$	$52.3_{\pm 3.1}$	$45.4_{\pm 2.1}$	$33.1_{\pm 5.0}$	$20.0_{\pm 5.9}$
S; X	$24.8_{\pm 4.1}$	$26.8_{\pm 3.4}$	$31.5_{\pm 5.6}$	$31.4_{\pm 5.9}$	$12.3_{\pm 6.2}$	$9.5_{\pm 2.8}$	$22.8_{\pm 5.8}$	$21.3_{\pm 4.8}$
C; G	$30.7_{\pm 3.2}$	$33.0_{\pm 4.4}$	$45.5_{\pm 4.0}$	$35.1_{\pm 1.4}$	$40.0_{\pm 7.5}$	$27.6_{\pm 13.8}$	$40.3_{\pm 0.6}$	$23.2_{\pm 6.6}$
C; E	$28.4_{\pm 2.1}$	$34.4_{\pm 1.9}$	$43.3_{\pm 4.5}$	$35.1_{\pm 2.9}$	$50.8_{\pm 3.8}$	$31.7_{\pm 3.9}$	$35.6_{\pm 4.6}$	$17.4_{\pm 7.8}$
C; B	$26.3_{\pm 2.8}$	$32.8_{\pm 1.5}$	$38.5_{\pm 3.2}$	$35.1_{\pm 2.5}$	$50.8_{\pm 3.8}$	$22.9_{\pm 4.1}$	$32.8_{\pm 7.0}$	$21.9_{\pm 3.8}$
C; X	$21.8_{\pm 4.1}$	$32.2_{\pm 3.1}$	$35.2_{\pm 5.2}$	$29.8_{\pm 3.3}$	$23.1_{\pm 8.4}$	$16.2_{\pm 2.7}$	$19.4_{\pm 4.7}$	$23.2_{\pm 7.7}$
H; G	$35.8_{\pm 3.1}$	$41.5_{\pm 4.0}$	$37.4_{\pm 4.4}$	$33.5_{\pm 2.3}$	$44.6_{\pm 5.8}$	$35.5_{\pm 10.1}$	$33.1_{\pm 5.4}$	$19.3_{\pm 3.5}$
H; E	$37.6_{\pm 4.0}$	$39.8_{\pm 1.3}$	$37.4_{\pm 5.9}$	$34.4_{\pm 5.2}$	$47.7_{\pm 5.8}$	$36.8_{\pm 7.2}$	$31.9_{\pm 6.5}$	$20.6_{\pm 8.3}$
H; B	$32.8_{\pm 3.0}$	$35.2_{\pm 3.3}$	$33.3_{\pm 3.7}$	$39.5_{\pm 3.0}$	$49.2_{\pm 6.2}$	$29.8_{\pm 8.0}$	$31.6_{\pm 7.3}$	$19.4_{\pm 8.4}$
H; X	$25.7_{\pm 4.3}$	$30.7_{\pm 3.3}$	$37.0_{\pm 4.5}$	$29.5_{\pm 1.6}$	$23.1_{\pm 10.9}$	$14.3_{\pm 5.9}$	$26.6_{\pm 4.4}$	$19.3_{\pm 5.4}$

S, C, H, G, E, B, X represent respectively ST-VQA, CoMem, HME-VQA, GloVe, ELMo, BERT, and XLM
We introduce "Act_{1st}", etc. in Table 1 to identify the question types

39.7% in PororoQA (Table 4). But, especially for EgoVQA, Table 3 shows that other techniques can provide useful embeddings depending on the architecture chosen: as an example, ST-VQA with GloVe achieves 35.7%, while HME-VQA with ELMo obtains 35.5%.

Table 6 Accuracy per question type over PororoQA using the frozen embeddings

M;E	Question type accuracy (%)										
	Abs	Act	Caus	Det	Loc	Met	Per	Reas	Stmt	Time	Y/N
S;G	$38.5_{\pm 1.7}$	$39.8_{\pm 2.0}$	$33.2_{\pm 3.9}$	$37.5_{\pm 3.7}$	$28.7_{\pm 4.6}$	$45.4_{\pm 1.9}$	$30.3_{\pm 0.9}$	$39.5_{\pm 4.3}$	$37.9_{\pm 2.8}$	$37.1_{\pm 4.1}$	$32.9_{\pm 3.3}$
S;E	$40.2_{\pm 2.2}$	$39.3_{\pm 1.6}$	$33.3_{\pm 5.2}$	$34.8_{\pm 4.1}$	$29.9_{\pm 5.7}$	$43.7_{\pm 2.5}$	$30.8_{\pm 1.1}$	$31.0_{\pm 5.5}$	$40.2_{\pm 1.9}$	$28.6_{\pm 5.4}$	$39.3_{\pm 3.0}$
S;B	$45.9_{\pm 2.7}$	$44.5_{\pm 1.7}$	$37.7_{\pm 5.7}$	$44.7_{\pm 7.0}$	$30.5_{\pm 4.1}$	$45.8_{\pm 1.7}$	$32.0_{\pm 0.5}$	$35.7_{\pm 2.5}$	$40.7_{\pm 2.9}$	$29.7_{\pm 4.7}$	$44.8_{\pm 1.8}$
S;X	$30.3_{\pm 1.8}$	$30.9_{\pm 1.5}$	$39.1_{\pm 4.5}$	$29.7_{\pm 4.7}$	$26.5_{\pm 4.2}$	$22.7_{\pm 2.2}$	$23.3_{\pm 1.7}$	$30.3_{\pm 2.6}$	$28.9_{\pm 1.3}$	$15.9_{\pm 2.9}$	$27.4_{\pm 2.2}$
C;G	$34.0_{\pm 2.4}$	$36.7_{\pm 1.5}$	$28.8_{\pm 1.8}$	$35.3_{\pm 6.3}$	$25.9_{\pm 2.6}$	$44.8_{\pm 3.5}$	$25.7_{\pm 2.4}$	$34.1_{\pm 3.5}$	$32.3_{\pm 4.7}$	$33.5_{\pm 5.1}$	$25.3_{\pm 2.8}$
C;E	$37.9_{\pm 5.9}$	$38.6_{\pm 1.3}$	$28.4_{\pm 7.1}$	$32.3_{\pm 3.7}$	$29.2_{\pm 2.4}$	$44.8_{\pm 2.2}$	$29.4_{\pm 1.0}$	$31.6_{\pm 6.6}$	$36.2_{\pm 4.6}$	$37.3_{\pm 6.7}$	$31.9_{\pm 2.6}$
C;B	$41.4_{\pm 1.4}$	$41.5_{\pm 1.7}$	$43.7_{\pm 2.7}$	$37.4_{\pm 7.0}$	$28.3_{\pm 2.3}$	$48.0_{\pm 4.2}$	$26.6_{\pm 0.4}$	$34.6_{\pm 3.8}$	$33.4_{\pm 1.1}$	$38.5_{\pm 5.6}$	$33.6_{\pm 5.1}$
C;X	$23.0_{\pm 3.4}$	$25.2_{\pm 1.1}$	$29.3_{\pm 6.9}$	$21.2_{\pm 3.5}$	$20.5_{\pm 2.5}$	$20.2_{\pm 2.6}$	$21.8_{\pm 1.9}$	$21.0_{\pm 2.8}$	$23.3_{\pm 1.8}$	$16.3_{\pm 1.9}$	$17.6_{\pm 4.6}$
H;G	$39.5_{\pm 4.2}$	$38.5_{\pm 1.8}$	$29.0_{\pm 4.6}$	$43.2_{\pm 3.6}$	$26.4_{\pm 3.9}$	$45.2_{\pm 3.5}$	$30.7_{\pm 1.5}$	$39.3_{\pm 4.0}$	$37.8_{\pm 2.2}$	$36.2_{\pm 8.3}$	$39.6_{\pm 4.2}$
H;E	$36.6_{\pm 3.7}$	$39.9_{\pm 1.0}$	$34.5_{\pm 11.4}$	$31.8_{\pm 4.4}$	$36.2_{\pm 6.2}$	$43.4_{\pm 2.5}$	$28.3_{\pm 1.5}$	$33.0_{\pm 1.4}$	$36.2_{\pm 3.1}$	$32.7_{\pm 6.6}$	$34.2_{\pm 6.9}$
H;B	$44.7_{\pm 2.9}$	$43.1_{\pm 1.7}$	$46.2_{\pm 2.8}$	$44.6_{\pm 6.2}$	$27.2_{\pm 3.2}$	$44.2_{\pm 2.0}$	$30.9_{\pm 2.1}$	$38.2_{\pm 2.4}$	$38.7_{\pm 2.1}$	$38.4_{\pm 6.2}$	$36.7_{\pm 1.3}$
H;X	$30.0_{\pm 1.4}$	$30.7_{\pm 1.9}$	$40.0_{\pm 6.6}$	$27.7_{\pm 4.7}$	$24.8_{\pm 4.4}$	$21.1_{\pm 2.4}$	$24.5_{\pm 1.9}$	$33.2_{\pm 2.3}$	$29.5_{\pm 2.3}$	$17.8_{\pm 1.3}$	$16.9_{\pm 4.5}$

Note that "*Abs*", etc. are introduced in Table 2 to identify the question types

To perform a finer-grained analysis, we present in Tables 5 and 6 the accuracy values based on the question type for EgoVQA and PororoQA. We perform this analysis because, as previously mentioned, question types may represent a key element when trying to answer a question. As an example, Table 5 shows that, when dealing with Act_{3rd} questions, HME-VQA with GloVe (row identified with “H;G”) achieves 41.5% average accuracy, yet when using BERT loses around 6% (row “H;B” shows 35.2%). Similarly, CoMem with GloVe (row “C;G”) has an accuracy of 40.3% when answering Cnt questions, yet it only obtains 32.8% when using BERT. Similar differences can be also observed on PororoQA in Table 6, e.g. when faced with $Reas$ questions, ST-VQA obtains 39.5% and 31.0% when adopting, respectively, GloVe (row “S;G”) or ELMo (row “S;E”). Thus, by analyzing the results while also taking the question type into account can lead to some insights which we detail in the next subsections.

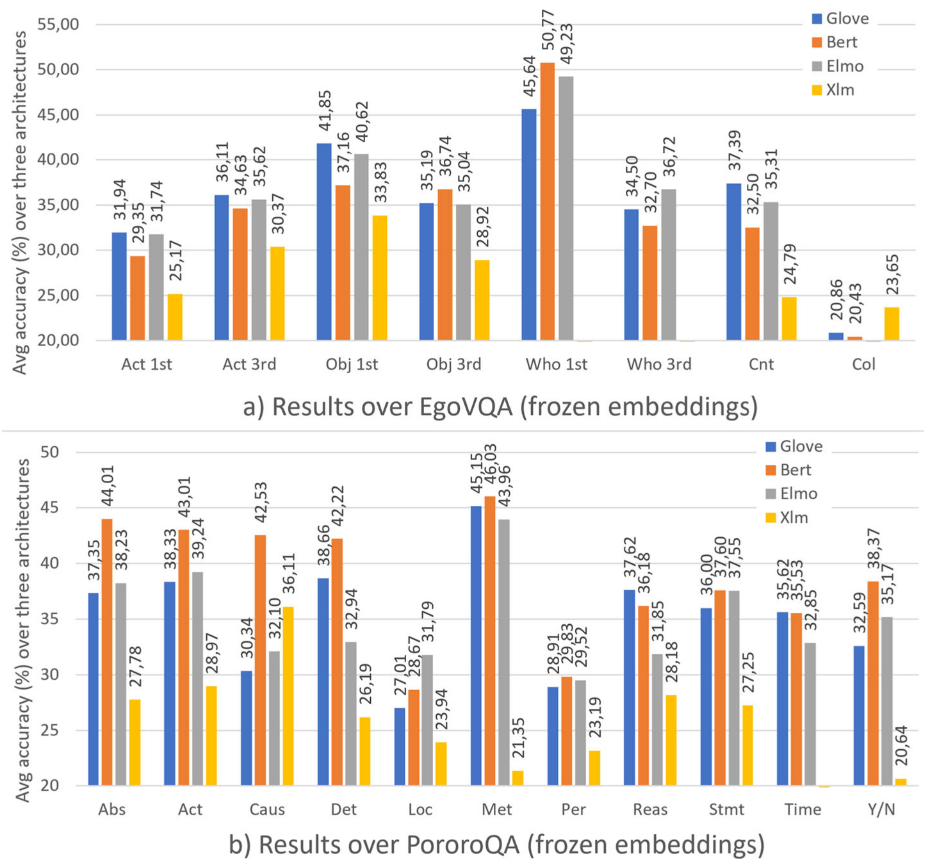


Fig. 5 We report for each question type the average accuracy (setting the minimum to the random chance, i.e. 20%) obtained by using a specific word embedding technique. It is possible to see that different question types are best dealt with by using different embeddings. Best viewed in color

4.4.1 EgoVQA

In Fig. 5 we propose two plots where we present the accuracy obtained when averaging with respect to the architecture used. For example, in Fig. 5a the blue column (related to GloVe) over Obj_{1st} shows 41.85, which is the mean value of the average accuracy obtained by ST-VQA, CoMem, and HME-VQA for that type of question. We do this in order to have a simplified view of the detailed results shown in Tables 5 and 6.

In the case of EgoVQA, Fig. 5a shows that on average GloVe and ELMo achieve better performance than BERT and XLM.

Moreover, we can also observe that ELMo obtains a 2.2% margin over GloVe when trying to identify who is performing a given action in front of the camera wearer (Who_{3rd}). Considering that the questions of this type, Who_{3rd}, are longer with respect to other types (11.5 words versus an average of 9.6), this may be due to ELMo having the memory capabilities provided by the LSTMs while also exploiting the bidirectional context. A similar reasoning could be applied to BERT as well: in fact, as shown in Table 5, when coupled with ST-VQA (row "S;B") it achieves the best result for this question type (45.4%), yet the mean value shown in Fig. 5a is lowered due to the low average and high variance obtained by the other two architectures (with CoMem, i.e. row "C;B", it obtains 22.9% while with HME-VQA, i.e. row "H;B", 29.8%).

Similarly, the question type Obj_{1st} is best tackled with GloVe embeddings (Fig. 5a reports 41.8% accuracy). In this case, around 78% of the questions are of the form "what am I holding", thus the long-term state provided by the LSTMs or by the Transformer encoder may be too complex to capture some of the information which, on the other hand, synergize well with the simplicity of GloVe.

4.4.2 PororoQA

Differently from EgoVQA, Fig. 5b shows that BERT is the overall best choice when dealing with PororoQA, although there are situations in which GloVe and ELMo perform comparably ($Stmt$, $Time$) or even better (Loc, $Reas$).

In this case, Table 6 is fundamental to detail some differences. HME-VQA coupled with ELMo performs better than all the other combinations for Loc questions: it achieves 36.2% accuracy (row "H;E"), while the second best is given by ST-VQA with BERT (row "S;B") which obtains 30.5% (hence, a 6% margin). Since the answers mainly differ due to nouns related to sceneries (e.g. "forest", "sea"), ELMo may be more effective at providing embeddings which cope better with these visual features. This proves extremely beneficial when coupled with the heterogeneous memory and the gating mechanisms exploited in HME-VQA, which help the model picking the correct association between the available multimodal features.

Obtaining "more than random" performance for $Stmt$ questions is also noteworthy because these questions involve the contents of a *speech*. Given that GloVe, ELMo, and BERT obtain around 37% accuracy as shown in Fig. 5b, several of these questions can be correctly answered to by only looking at the RGB frames, the question text, and the answer text. As an example of this, the possible answers for the question "what did Poby and his friend say" are permutations of "paper rock scissor": the models thus learn how to correctly answer by extracting spatio-temporal visual features which lead them to correctly identify the three hand gestures in the clip.

Another interesting detail can be seen in the $Caus$ question type, which involves questions asking to identify an event which happened in relation to another one (e.g. "what

Table 7 Average accuracy (with absolute and average changes wrt Table 3) over EgoVQA after the finetuning of the embeddings

	GloVe*	ELMo*	BERT*	XLM*	*avg*
ST-VQA	$35.7_{\pm 4.1}$(+0.0)	$34.0_{\pm 4.7}$(− 0.9)	$34.9_{\pm 4.9}$(− 1.2)	$25.1_{\pm 4.2}$(+1.1)	− 0.2
CoMem	$33.9_{\pm 3.7}$(− 0.2)	$32.4_{\pm 5.0}$(− 1.2)	$33.1_{\pm 4.7}$(+1.5)	$22.3_{\pm 4.2}$(− 3.4)	− 0.8
HME-VQA	$35.6_{\pm 3.2}$(+0.2)	$34.9_{\pm 3.8}$(-0.6)	$36.4_{\pm 4.4}$(+2.8)	$25.9_{\pm 4.0}$(-0.9)	+0.4
avg	+*0.0*	− 0.9	+*1.0*	− 1.1	− 0.2

happened when the egg broke?", "a green little dinosaur popped out"). The best result is obtained by BERT (Fig. 5b reports around 42.5% accuracy), but XLM manages to shine as well (36.1%) obtaining a margin of at least 4% over ELMo (32.1%) and GloVe (30.3%). This is likely due to two facts: Transformer-based embeddings, and bidirectionality. The first point could be due to both the depth of the network and the attention mechanisms exploited in the Transformer, which are not used in GloVe nor in ELMo. The second point is supported by noticing that these questions are likely better understood when read both directions (the event described in the answer might have happened before the one described in the question, or vice versa), and by the fact that ELMo exhibits this property as well, leading it to be more accurate (around +2%) than GloVe.

4.5 Finetuning the embeddings

As a second set of experiments, we focus on the finetuning step of the embedding modules. This is usually performed because it helps the model gain a considerable boost, since it helps rearranging the embedding space in a way to make the features more related to the task at hand. As an example, GloVe embeddings are finetuned in [6].

Instead of starting the training from scratch, we restart from the weights learned during the previous step. The procedure we follow can be described as such: first of all, we select the model with the best validation accuracy and use it to set the initial weights; then, we freeze all the components (but the embedding module) and perform the training for 20 epochs using a fixed learning rate of 5e-5. We repeat this procedure five times and we also use the same seed, i.e. when performing the i-th iteration of this procedure, we fix seed i and use the weights that were computed using seed i.

Table 8 Average accuracy (with absolute and average changes wrt Table 4) over PororoQA after the finetuning of the embeddings

	GloVe*	ELMo*	BERT*	XLM*	*avg*
ST-VQA	$36.5_{\pm 0.5}$(− 0.3)	$38.2_{\pm 1.0}$(+2.5)	$41.2_{\pm 1.0}$(+1.5)	$28.7_{\pm 0.8}$(+1.3)	+*1.2*
CoMem	$35.3_{\pm 0.8}$(+2.0)	$37.0_{\pm 1.5}$(+1.6)	$33.0_{\pm 7.2}$(− 3.1)	$29.1_{\pm 0.4}$(+6.9)	+*1.9*
HME-VQA	$37.6_{\pm 1.0}$(+0.8)	$39.1_{\pm 1.9}$(+4.6)	$43.5_{\pm 2.0}$(+4.3)	$29.3_{\pm 0.7}$(+1.5)	+*2.8*
avg	+*0.8*	+*2.9*	+*0.9*	+*3.2*	+*2.6*

4.5.1 EgoVQA

From Table 7 it can be noted that, although the best result has improved, the finetuning procedure often does not help, e.g. with ELMo (on average, it loses around 0.9%). This is likely due to the dataset being too small to benefit from the finetuning, leading almost all the models to overfit. Yet, it can be seen that BERT benefits the most from this procedure (on average +1.0%) and, in particular, HME-VQA with BERT obtains a +2.8% improvement (+3.1% wrt the best result published in [5]).

4.5.2 PororoQA

Differently from EgoVQA, finetuning is particularly helpful and beneficial over PororoQA. Table 8 reports an average improvement of 2.6%, with a peak of +6.9% when finetuning XLM using CoMem. Table 9 shows a less varied situation than Table 6, with BERT being the overall best choice. Yet, some interesting results may be distilled from it.

First of all, after the finetuning step XLM achieves the best accuracy when dealing with the question type $Caus$ (obtaining 45.2% when using CoMem). Although BERT obtains a lower average accuracy with respect to the previous performance (likely due to the learning rate being too high), BERT and XLM achieve 37.3% and 41.4% (Table 8), when averaging with respect to the architecture. Considering that GloVe and ELMo achieve 26.9% and 31.3%, this may confirm the previous hypothesis involving bidirectionality and network depth.

Secondly, ELMo receives on average a 5.3% improvement (from an average of 32.8% in Table 6 to 38.1% in Table 8, computed with respect to the three architectures) when tackling $Time$ questions. Considering that these questions often involve reasoning about temporally-related events, the bidirectionality of ELMo coupled with the LSTM gating and memory capabilities may be the key point which helps understanding these relations.

Table 9 Accuracy per question type over PororoQA after the finetuning of the embeddings

M;E	Question type accuracy (%)										
	Abs	Act	Caus	Det	Loc	Met	Per	Reas	Stmt	Time	Y/N
S;G*	$37.2_{\pm1.2}$	$42.5_{\pm1.5}$	$27.4_{\pm3.7}$	$35.7_{\pm1.9}$	$24.7_{\pm3.4}$	$45.4_{\pm1.7}$	$34.6_{\pm3.0}$	$36.8_{\pm5.7}$	$40.7_{\pm2.1}$	$38.6_{\pm3.5}$	$30.2_{\pm2.0}$
S;E*	$39.3_{\pm4.8}$	$41.1_{\pm1.9}$	$35.8_{\pm5.9}$	$38.8_{\pm6.3}$	$30.8_{\pm1.4}$	$46.2_{\pm3.2}$	$34.1_{\pm3.8}$	$35.0_{\pm3.2}$	$44.4_{\pm1.9}$	$34.0_{\pm5.6}$	$42.7_{\pm2.5}$
S;B*	$43.3_{\pm4.4}$	$43.3_{\pm1.7}$	$36.5_{\pm7.7}$	$43.0_{\pm7.1}$	$33.1_{\pm4.7}$	$52.9_{\pm3.2}$	$33.9_{\pm2.4}$	$40.7_{\pm3.1}$	$43.0_{\pm3.1}$	$31.9_{\pm6.3}$	$48.6_{\pm4.7}$
S;X*	$28.8_{\pm0.9}$	$34.8_{\pm1.8}$	$37.4_{\pm3.0}$	$30.2_{\pm4.2}$	$23.9_{\pm2.1}$	$20.9_{\pm2.9}$	$25.0_{\pm1.6}$	$34.4_{\pm0.4}$	$28.4_{\pm1.6}$	$22.3_{\pm7.7}$	$17.1_{\pm2.4}$
C;G*	$34.1_{\pm2.9}$	$39.0_{\pm2.1}$	$25.7_{\pm2.3}$	$40.4_{\pm5.7}$	$23.2_{\pm4.7}$	$43.0_{\pm3.4}$	$27.2_{\pm2.5}$	$34.9_{\pm2.8}$	$39.2_{\pm4.6}$	$37.8_{\pm6.4}$	$33.8_{\pm2.8}$
C;E*	$38.0_{\pm2.1}$	$38.2_{\pm2.4}$	$26.0_{\pm4.3}$	$36.2_{\pm5.1}$	$32.5_{\pm6.2}$	$41.1_{\pm5.0}$	$32.0_{\pm2.4}$	$35.5_{\pm4.4}$	$42.0_{\pm3.1}$	$38.9_{\pm4.3}$	$37.7_{\pm2.1}$
C;B*	$34.0_{\pm7.5}$	$35.6_{\pm8.6}$	$39.1_{\pm3.4}$	$34.1_{\pm5.9}$	$30.5_{\pm7.0}$	$31.8_{\pm10.2}$	$27.9_{\pm6.7}$	$36.3_{\pm8.2}$	$33.6_{\pm5.9}$	$27.0_{\pm7.1}$	$28.3_{\pm14.8}$
C;X*	$33.4_{\pm2.9}$	$33.5_{\pm1.6}$	$45.2_{\pm2.6}$	$32.1_{\pm1.6}$	$25.2_{\pm3.7}$	$21.5_{\pm1.6}$	$26.1_{\pm2.3}$	$33.5_{\pm2.2}$	$29.7_{\pm1.3}$	$18.5_{\pm2.5}$	$21.5_{\pm1.5}$
H;G*	$38.3_{\pm2.9}$	$37.8_{\pm0.8}$	$27.8_{\pm3.1}$	$43.1_{\pm2.3}$	$26.0_{\pm3.4}$	$41.1_{\pm2.7}$	$33.7_{\pm1.4}$	$42.4_{\pm2.3}$	$39.3_{\pm2.2}$	$38.1_{\pm7.5}$	$38.4_{\pm1.5}$
H;E*	$40.4_{\pm4.1}$	$39.9_{\pm2.4}$	$32.0_{\pm4.1}$	$38.5_{\pm3.8}$	$34.4_{\pm3.5}$	$49.1_{\pm4.2}$	$32.5_{\pm1.8}$	$36.6_{\pm2.6}$	$41.8_{\pm1.3}$	$41.4_{\pm3.5}$	$39.7_{\pm2.6}$
H;B*	$43.6_{\pm3.6}$	$43.6_{\pm2.1}$	$36.4_{\pm1.4}$	$49.7_{\pm5.5}$	$34.9_{\pm4.8}$	$50.9_{\pm3.8}$	$34.6_{\pm2.3}$	$45.1_{\pm3.4}$	$45.8_{\pm2.1}$	$33.8_{\pm8.1}$	$46.4_{\pm5.5}$
H;X*	$32.7_{\pm0.6}$	$33.4_{\pm1.3}$	$41.5_{\pm5.3}$	$31.6_{\pm3.8}$	$27.0_{\pm2.7}$	$19.3_{\pm1.9}$	$22.8_{\pm2.9}$	$36.6_{\pm1.4}$	$29.5_{\pm1.8}$	$26.5_{\pm4.6}$	$17.4_{\pm1.8}$

Table 10 Average accuracy (with absolute and average changes wrt Table 3) over EgoVQA after the adoption of the multi-task learning strategy (frozen embeddings)

	GloVe	ELMo	BERT	XLM	*avg*
ST-VQA	$\mathbf{36.5}_{\pm\mathbf{4.7}}$(+0.8)	$33.7_{\pm 5.0}$(− 1.2)	$\mathbf{36.2}_{\pm\mathbf{6.9}}$(+0.1)	$25.2_{\pm 3.1}$(+1.2)	*+0.2*
CoMem	$32.9_{\pm 3.2}$(− 1.2)	$33.8_{\pm 4.3}$(+0.2)	$31.2_{\pm 4.5}$(− 0.4)	$24.0_{\pm 3.1}$(− 1.7)	− 0.8
HME-VQA	$34.5_{\pm 3.4}$(− 0.9)	$35.1_{\pm 4.0}$(− 0.4)	$35.0_{\pm 5.5}$(+1.4)	$27.2_{\pm 3.9}$(+0.4)	+0.1
avg	− 0.4	− 0.6	*+0.3*	− 0.5	− 0.2

Although generally smaller, an improvement over *Time* questions is also observed with XLM and BERT.

Finally, HME-VQA coupled with BERT achieves both the best overall result (43.5% in Table 8) and also the best result over several question types. While this is partially due to BERT and its abilities to compute semantically rich embeddings, it surely confirms that HME-VQA is a powerful model able to capture multimodal cues which makes it great for VideoQA [6].

4.6 Adoption of multi-task learning strategy

As can be seen from the previous experiments, different word embedding techniques perform differently depending on the question type under analysis. This result is likely related to the different embedding techniques being able to capture some patterns in the question which depend on the type and are helpful to localize the answer within the video. To prove this surmise, we propose to adopt a multi-task learning strategy (detailed in Section 3.1), and show that it helps the models achieve a better generalization.

4.6.1 EgoVQA

Table 10 shows that HME-VQA coupled with BERT is the combination which benefits the most from the adoption of the multi-task strategy, gaining on average 1.4% accuracy. Yet, several of the other combinations gain only marginal improvements or even obtain a lower accuracy. This is likely due to the models overfitting even more than before, due to the added parameters.

Nonetheless, there are also other combinations which benefit from the added parameters as well. In particular, it can be noted that ST-VQA synergizes the best with the proposed technique, since it improves its performance when coupled with GloVe (+0.8%) and XLM

Table 11 Average accuracy (with absolute and average changes wrt Table 4) over PororoQA after the adoption of the multi-task learning strategy (frozen embeddings)

	GloVe	ELMo	BERT	XLM	*avg*
ST-VQA	$37.3_{\pm 1.4}$(+0.5)	$36.6_{\pm 1.1}$(+0.9)	$\mathbf{40.6}_{\pm\mathbf{1.5}}$(+0.9)	$28.7_{\pm 1.5}$(+1.3)	*+0.9*
CoMem	$35.0_{\pm 1.1}$(+1.7)	$36.1_{\pm 0.8}$(+0.7)	$37.0_{\pm 1.6}$(+0.9)	$24.0_{\pm 0.7}$(+1.8)	*+1.3*
HME-VQA	$35.8_{\pm 1.2}$(− 1.0)	$36.8_{\pm 0.5}$(+2.3)	$38.7_{\pm 1.3}$(− 0.5)	$27.9_{\pm 1.1}$(+0.1)	*+0.2*
avg	*+0.4*	*+1.3*	*+0.4*	*+1.1*	*+1.0*

Table 12 Accuracy per question type over PororoQA after the adoption of the multi-task learning strategy (frozen embeddings)

M;E	Question type accuracy (%)										
	Abs	Act	Caus	Det	Loc	Met	Per	Reas	Stmt	Time	Y/N
S;G	$38.8_{\pm 2.8}$	$39.9_{\pm 2.8}$	$33.3_{\pm 5.7}$	$39.3_{\pm 3.6}$	$31.2_{\pm 3.9}$	$47.9_{\pm 3.0}$	$30.0_{\pm 1.6}$	$38.5_{\pm 2.5}$	$42.4_{\pm 2.9}$	$39.5_{\pm 4.9}$	$30.3_{\pm 5.1}$
S;E	$38.0_{\pm 1.0}$	$39.7_{\pm 2.0}$	$24.3_{\pm 4.0}$	$34.7_{\pm 0.7}$	$30.6_{\pm 4.4}$	$49.3_{\pm 2.6}$	$31.0_{\pm 1.5}$	$34.9_{\pm 5.4}$	$41.6_{\pm 3.9}$	$38.5_{\pm 6.2}$	$39.6_{\pm 5.3}$
S;B	$43.6_{\pm 1.6}$	$42.2_{\pm 0.9}$	$39.0_{\pm 7.1}$	$43.6_{\pm 5.2}$	$30.3_{\pm 2.8}$	$46.3_{\pm 2.9}$	$35.3_{\pm 1.7}$	$34.0_{\pm 4.5}$	$41.2_{\pm 3.3}$	$33.5_{\pm 4.0}$	$42.9_{\pm 4.0}$
S;X	$28.9_{\pm 2.3}$	$30.5_{\pm 2.0}$	$45.0_{\pm 2.2}$	$34.7_{\pm 3.5}$	$22.5_{\pm 4.6}$	$21.6_{\pm 1.6}$	$24.9_{\pm 1.2}$	$33.7_{\pm 2.1}$	$29.9_{\pm 2.1}$	$20.0_{\pm 4.7}$	$21.0_{\pm 1.5}$
C;G	$36.9_{\pm 2.6}$	$38.6_{\pm 1.9}$	$33.6_{\pm 5.6}$	$41.0_{\pm 4.7}$	$26.3_{\pm 2.4}$	$41.7_{\pm 3.2}$	$28.3_{\pm 3.9}$	$34.5_{\pm 4.3}$	$34.8_{\pm 3.2}$	$36.9_{\pm 4.1}$	$34.8_{\pm 3.3}$
C;E	$40.0_{\pm 2.1}$	$40.2_{\pm 1.9}$	$29.7_{\pm 7.9}$	$32.2_{\pm 2.7}$	$29.2_{\pm 3.9}$	$45.5_{\pm 2.2}$	$29.9_{\pm 1.8}$	$36.9_{\pm 3.1}$	$39.8_{\pm 3.5}$	$28.6_{\pm 7.3}$	$36.5_{\pm 4.3}$
C;B	$42.8_{\pm 1.9}$	$39.3_{\pm 2.2}$	$38.0_{\pm 8.8}$	$39.5_{\pm 4.4}$	$29.8_{\pm 3.9}$	$46.0_{\pm 2.8}$	$26.2_{\pm 2.4}$	$36.2_{\pm 1.8}$	$37.5_{\pm 3.5}$	$34.4_{\pm 5.0}$	$34.9_{\pm 3.8}$
C;X	$23.5_{\pm 4.2}$	$24.3_{\pm 3.8}$	$31.4_{\pm 2.6}$	$23.6_{\pm 4.2}$	$22.1_{\pm 2.3}$	$24.2_{\pm 2.6}$	$23.3_{\pm 1.5}$	$22.4_{\pm 4.0}$	$23.3_{\pm 2.4}$	$11.5_{\pm 2.5}$	$23.0_{\pm 4.6}$
H;G	$38.8_{\pm 2.5}$	$39.2_{\pm 0.9}$	$36.7_{\pm 4.7}$	$41.6_{\pm 4.6}$	$27.6_{\pm 2.5}$	$45.4_{\pm 3.6}$	$29.1_{\pm 2.4}$	$37.7_{\pm 6.1}$	$41.8_{\pm 1.3}$	$23.9_{\pm 5.6}$	$32.7_{\pm 3.4}$
H;E	$38.9_{\pm 2.1}$	$41.2_{\pm 0.9}$	$26.2_{\pm 6.2}$	$36.1_{\pm 5.0}$	$34.3_{\pm 4.2}$	$44.9_{\pm 4.1}$	$32.1_{\pm 1.1}$	$30.2_{\pm 4.6}$	$42.2_{\pm 3.6}$	$35.0_{\pm 4.7}$	$40.0_{\pm 5.7}$
H;B	$44.4_{\pm 1.2}$	$42.0_{\pm 3.1}$	$41.5_{\pm 8.3}$	$40.3_{\pm 2.0}$	$26.4_{\pm 1.0}$	$45.7_{\pm 4.7}$	$32.1_{\pm 2.9}$	$36.9_{\pm 4.3}$	$43.7_{\pm 4.9}$	$31.0_{\pm 3.8}$	$38.4_{\pm 4.6}$
H;X	$29.5_{\pm 2.5}$	$32.3_{\pm 1.3}$	$43.6_{\pm 7.7}$	$26.2_{\pm 1.7}$	$19.7_{\pm 2.5}$	$22.4_{\pm 1.0}$	$25.9_{\pm 2.2}$	$34.8_{\pm 2.8}$	$31.1_{\pm 2.3}$	$20.0_{\pm 6.4}$	$25.2_{\pm 1.9}$

(+1.2%) embeddings. This may be due to its simplicity and lower amount of parameters with respect to CoMem and HME-VQA.

To understand whether the difference in performance before and after the addition of the proposed multi-task learning strategy is significant, we use the Almost Stochastic Order (ASO) test [2, 4], as implemented by [47]. This test operates on the cumulative distribution function of the two models (before and after training with the proposed strategy) and estimates the amount of violation of the stochastic order. It formulates the following null hypothesis: $H_0 : \epsilon_{\min} \geq \tau$, which can be interpreted as the standard training being stochastically dominant in more cases than the training performed with the proposed strategy. To reject this hypothesis, $\epsilon_{\min} < \tau$ where $\tau = 0.5$. Using ASO with a confidence level $\alpha = 0.05$ it was possible to confirm some of the results we observed. In particular we found that, based on five random seeds, ST-VQA with ELMo, CoMem paired with GloVe or XLM, and HME-VQA paired with GloVe are stochastically dominant (in particular, $\epsilon_{\min} \geq 0.80$) if trained without the proposed strategy; in the cases of HME-VQA with ELMo or XLM, CoMem with ELMo or BERT, and ST-VQA with BERT the $\epsilon_{\min}$ is close to the threshold τ, so the difference is not as significant. In the other cases, the addition of the multi-task learning strategy leads to stochastically dominant solutions, with $\epsilon_{\min} \leq 0.40$ in most cases.

4.6.2 PororoQA

As can be seen in Table 11, almost all the different combinations of architectures and embeddings benefit from the adoption of the multi-task learning strategy: overall, if we compare to the results obtained before the finetuning of the embeddings (Table 4, changes are reported in Table 11), the improvement obtained by adopting the proposed strategy amounts to around +1.0% with a peak of +2.3% when using HME-VQA with ELMo. This shows that such simple addition helps the models both generalize better and understand what they should focus on based on the type of the question.

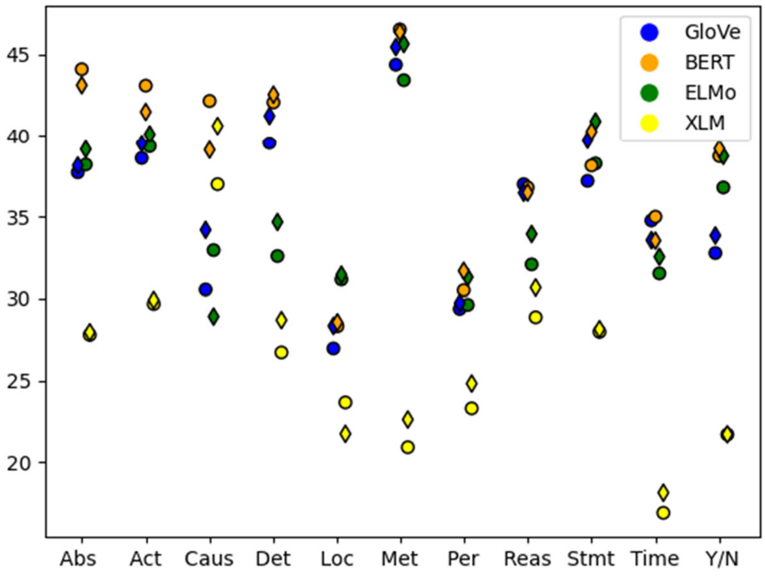


Fig. 6 Average accuracy over PororoQA, averaged with respect to the three architectures, before (○) and after (◊) the adoption of the proposed multi-task learning strategy. Best viewed in color

To ease the comparison of Tables 6 and 12, we propose in Fig. 6 a simplified view of the results per question type, where we visualize the average accuracy obtained by the embedding techniques (mean values with respect to the three architectures) before and after the adoption of the multi-task learning strategy. While it shows an overall improvement, such a figure also shows there are situations in which the proposed technique is beneficial or not based on the embedding technique used. As an example, for $Caus$ questions, GloVe and XLM benefit greatly from the additional task, whereas BERT and ELMo do not. More in detail, Table 12 shows that a considerable boost (+9.9%) is obtained by ST-VQA coupled with ELMo when dealing with $Time$ questions, where the accuracy goes from 28.6% (in Table 6, row "S;E") to 38.5%. Since a similar improvement was also obtained when finetuning ELMo for this question type, this further strengthens the hypothesis previously formulated.

As in the previous case, we use ASO to determine the significance of the results. With a confidence level $\alpha = 0.05$, we found that the addition of the proposed multi-task learning strategy leads to solutions which are stochastically dominant over the model trained without

Table 13 The average time (in seconds) spent per sample at inference time for each of the 12 combinations considered in this manuscript

	Embedding technique				
Architecture	GloVe	ELMo	BERT	XLM	*avg*
ST-VQA	0.005	0.039	0.015	0.026	0.021
CoMem	0.064	0.101	0.074	0.095	0.083
HME-VQA	0.050	0.085	0.062	0.081	0.069
avg	0.040	0.075	0.050	0.067	-

Table 14 Average accuracy (with absolute and average changes wrt Table 11) over PororoQA after the adoption of the multi-task learning strategy and the finetuning of the embeddings

	GloVe*	ELMo*	BERT*	XLM*	*avg*
ST-VQA	$37.8_{\pm 1.2}$(+0.5)	$38.8_{\pm 0.9}$(+2.2)	$42.7_{\pm 1.4}$(+2.1)	$30.0_{\pm 0.6}$(+1.3)	*+1.5*
CoMem	$35.6_{\pm 0.6}$(+0.6)	$37.8_{\pm 1.2}$(+1.7)	$42.7_{\pm 1.8}$(+5.7)	$28.7_{\pm 1.2}$(+4.7)	*+3.2*
HME-VQA	$37.3_{\pm 0.9}$(+1.5)	$38.2_{\pm 1.2}$(+1.4)	$42.6_{\pm 1.1}$(+3.9)	$28.9_{\pm 0.6}$(+1.0)	*+1.9*
avg	*+0.9*	*+1.8*	*+3.9*	*+2.3*	*+2.9*

the proposed strategy in most of the cases (with $\epsilon_{\min} < 0.40$). In the case of HME-VQA, the addition of the proposed strategy leads to a stochastically dominant solution only when paired with ELMo ($\epsilon_{\min} < 0.10$) whereas, according to the statistical test, the model trained without the proposed strategy is either stochastically dominant (with GloVe and BERT, $\epsilon_{\min} \geq 0.85$) or the same as the model trained with the proposed strategy (with XLM, $\epsilon_{\min} = 0.5$) (Table 13).

4.7 Embeddings finetune after the multi-task learning

As we did previously, we start from the weights obtained during the training with the multi-task learning strategy and proceed with the finetuning of the embeddings alone. Overall, a greatly positive outcome is achieved for PororoQA (Table 14), whereas over EgoVQA it does not help at all.

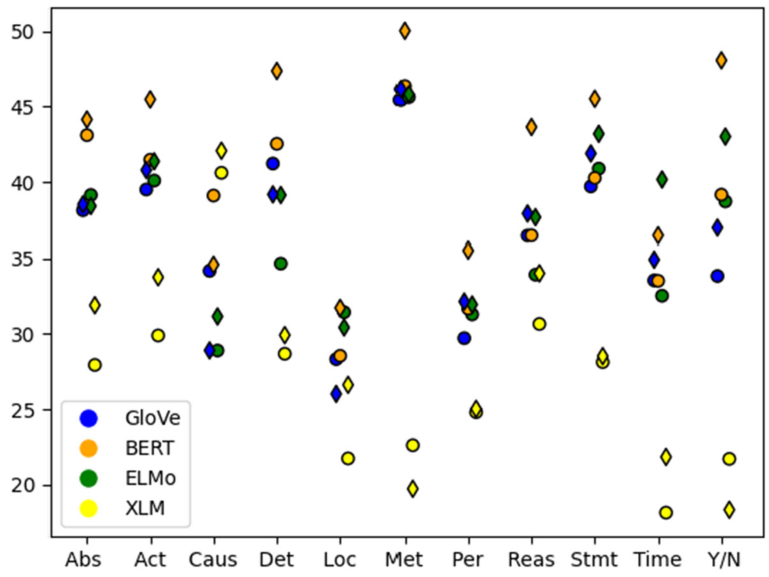


Fig. 7 Accuracy over PororoQA obtained by the embedding techniques (average wrt models trained with the proposed multi-task learning strategy) before (○) and after (◇) finetuning. Best viewed in color

4.7.1 PororoQA

Table 14 reports on average a +2.9% improvement, with even higher peaks when using CoMem (+5.7%). Also in this case we propose a simplified view of the comparison between Tables 12 and 14 in Fig. 7, where we visualize the accuracy obtained before and after finetuning. From the Figure it can be seen that BERT and ELMo always benefit from the finetuning procedure. In particular, from Table 14 it can be seen that the average improvement amounts to 1.8% for ELMo and 3.9% for BERT. It follows that, generally speaking, it is a wise choice to finetune the embeddings, especially when there is a decent amount of available data.

4.8 About inference times

In this section, we analyze the time taken to predict the answer during the inference phase. In particular, the total time required by the pipeline analyzed by isolating the feature extraction of the video from all the other operations. This is done because the visual feature extraction is performed in the same way for all the considered solutions. In fact, all of them, as previously described in Section. 3, use VGG and C3D which, on average, take less than 150 ms per video clip. Therefore, the time taken by this step can be removed from the total time in order to make it clear the overhead taken by the specific architecture or word embedding techniques. Conversely, the extraction of the textual features is tightly linked to the word embedding technique used. In Table 13 we report the average time taken by all the combinations of overall model (ST-VQA, CoMem, or HME-VQA) and word embedding technique considered in this study. According to this analysis, ST-VQA combined with GloVe represents the fastest solution taking only 5 ms on average. In particular, GloVe represents the fastest word embedding approach since it only needs to map tokens to vectors through a table, whereas BERT, ELMo, and XLM have additional layers which slow down the process, leading respectively to 50, 75, and 67 ms on average. Moreover, since it is the simplest architecture considered in this study, ST-VQA is also the fastest among the three (on average, 21 ms compared to 83 and 69 ms taken by CoMem and HME-VQA).

4.9 Take-home messages

Word embeddings Each of the analyzed embedding techniques deals better with specific question types, likely implying questions have characteristics which are encoded differently (and possibly ignored) by each technique. Over EgoVQA, ELMo works better when identifying an actor (Who_{3rd}), and GloVe is effective when identifying objects (Obj_{1st}); over PororoQA, ELMo performs significantly better than GloVe, XLM, and BERT when identifying locations (Loc), and the synergy between the bidirectionality and the Transformer-based encoder, used by XLM and BERT, is beneficial when guessing which event happened in relation to another one ($Caus$).

Importance of the embedding choice In relation to the previous message, we thoroughly showed that the choice of the embedding technique to use should take into account which question type (and the properties of its questions) is the most prevalent in the considered dataset.

Bidirectionality We provide evidence showing that bidirectionality is convenient when both Q and A are rich and complex sentences (e.g. $Caus$, $Time$ questions). Although for the latter it becomes clearer when finetuning, for the former it is noticeable also when using the frozen embeddings.

Finetuning Although the improvements due to finetuning are harder to see with EgoVQA due to its smaller size, it is diaphanous for PororoQA: finetuning helps rearranging the embedding space, making it easier for the models to understand and link the textual and the visual concepts.

Multi-task learning Question types raise the possibility to perform finer-grained analysis, but they can also be exploited to achieve improved generalization. We show this is possible by proposing a multi-task learning strategy which takes question types into account.

5 Conclusion

VideoQA has recently seen a surge of interest thanks to the release of several rich and public datasets. In VideoQA, to provide a meaningful answer, the model needs to understand both the visual and the textual content. Given the multitude of word embedding techniques and considering that the computed representations influence the final performance of the VideoQA model, we explore the use of several of them on two public datasets: EgoVQA and PororoQA. We find that the embeddings computed by BERT are the best overall solution, but we also find that depending on the question type different embeddings should be preferred.

Moreover, we showed this result can be further exploited by introducing a multi-task learning approach where the models are also asked to classify the given questions: a simple yet effective technique which greatly helps the overall performance of the considered solutions.

Finally, we show that more accurate predictions can be obtained by finetuning the embeddings, both with and without the proposed multi-task learning strategy. BERT is the technique benefiting the most from it, but there are situations in which the improvement can be substantially large when taking into account specific question types, e.g. the synergy between ELMo and $Time$ questions. At the end of the experimental section we also collect some “take-home messages” (Section 4.9) where we summarize the main results observed in this study.

As a future work, several other word embedding techniques can be tested, such as DistilBERT [38] and RoBERTa [29]. Moreover, we focused on EgoVQA and PororoQA, but these results should help obtaining improved performance in several other datasets, such as TGIF-QA [15] or MovieQA [45], where it is possible to define the type of the questions. In particular, automatically identifying the type of the question may be an interesting research direction. The types may be defined by a rule-based system (e.g. inspired by the “five Ws”), or by using clustering techniques to automatically discover clusters of semantically related questions. Furthermore, in our multi-task learning approach we focused on a single auxiliary task designed on the concept of question type, but additional tasks may be used to extend it and help the model extracting more general features. Finally, the VideoQA community is mostly focusing on defining new methods to achieve better performance. Nonetheless, predicting the correct answer with a lower time delay may have important consequences on several applications, therefore it may become an interesting research direction.

Acknowledgements We gratefully acknowledge the support from Amazon AWS Machine Learning Research Awards (MLRA) and NVIDIA AI Technology Centre (NVAITC), EMEA. We acknowledge the CINECA award under the ISCRA initiative, which provided computing resources for this work.

Funding Open access funding provided by Università degli Studi di Udine within the CRUI-CARE Agreement. No funding was received for conducting this study.

Data Availability The datasets analyzed during the current study are publicly available, and are also available from the corresponding author on reasonable request.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: ICLR
2. Del Barrio E, Cuesta-Albertos JA, Matrán C (2018) An optimal transportation approach for assessing almost stochastic order. In: The mathematics of the uncertain. Springer, p 33–44
3. Devlin J, Chang MW, Lee K et al (2019) Bert: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT
4. Dror R, Shlomov S, Reichart R (2019) Deep dominance - how to properly compare deep neural models. In: Korhonen A, Traum DR, Màrquez L (eds) Proceedings of the 57th conference of the association for computational linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1 : Long Papers. Association for Computational Linguistics, pp 2773–2785. https://doi.org/10.18653/v1/p19-1266
5. Fan C (2019) Egovqa - an egocentric video question answering benchmark dataset. In: ICCV Workshop
6. Fan C, Zhang X, Zhang S et al (2019) Heterogeneous memory enhanced multimodal attention model for video question answering. In: CVPR
7. Fang Z, Liu J, Li Y et al (2019) Improving visual question answering using dropout and enhanced question encoder. Pattern Recogn 90:404–414
8. Gao J, Ge R, Chen K et al (2018) Motion-appearance co-memory networks for video question answering. In: CVPR
9. Garcia N, Otani M, Chu C et al (2020) Knowit vqa: answering knowledge-based questions about videos. In: Proceedings of the AAAI conference on artificial intelligence, pp 10,826–10,834
10. Gardner M, Grus J, Neumann M et al (2017) Allennlp: a deep semantic natural language processing platform. arXiv:180307640
11. He K, Zhang X, Ren S et al (2016) Deep residual learning for image recognition. In: CVPR
12. Hori C, Hori T, Lee TY et al (2017) Attention-based multimodal fusion for video description. In: ICCV
13. Huang D, Chen P, Zeng R et al (2020) Location-aware graph convolutional networks for video question answering. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 11,021–11,028
14. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, PMLR, pp 448–456
15. Jang Y, Song Y, Yu Y et al (2017) Tgif-qa: toward spatio-temporal reasoning in visual question answering. In: CVPR
16. Jiang P, Han Y (2020) Reasoning with heterogeneous graph alignment for video question answering. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 11,109–11,116
17. Karpathy A, Toderici G, Shetty S et al (2014) Large-scale video classification with convolutional neural networks. In: CVPR

18. Kim J, Ma M, Kim K et al (2019) Gaining extra supervision via multi-task learning for multi-modal video question answering. In: International Joint Conference on Neural Networks (IJCNN). IEEE, pp 1-8
19. Kim KM, Heo MO, Choi SH et al (2017) Deepstory: video story qa by deep embedded memory networks. In: IJCAI
20. Kim KM, Choi SH, Kim JH et al (2018) Multimodal dual attention memory for video story question answering. In: ECCV
21. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: ICLR
22. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. ICLR
23. Krishna R, Zhu Y, Groth O et al (2017) Visual genome: connecting language and vision using crowdsourced dense image annotations. Int J Comput Vis 123(1):32–73
24. Lample G, Conneau A (2019) Cross-lingual language model pretraining. In: NeurIPS
25. Lei J, Yu L, Bansal M et al (2018) Tvqa: localized, compositional video question answering. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp 1369–1379
26. Lei J, Li L, Zhou L et al (2021) Less is more: clipbert for video-and-language learning via sparse sampling. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7331–7341
27. Li L, Chen YC, Cheng Y et al (2020) Hero: hierarchical encoder for video+ language omni-representation pre-training. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 2046–2065
28. Li X, Song J, Gao L et al (2019) Beyond rnns: positional self-attention with co-attention for video question answering. In: Proceedings of the AAAI conference on artificial intelligence, pp 8658–8665
29. Liu Y, Ott M, Goyal N et al (2019) Roberta: a robustly optimized bert pretraining approach. arXiv:190711692
30. Lu J, Goswami V, Rohrbach M et al (2020) 12-in-1: multi-task vision and language representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10,437–10,446
31. Luo Y, Zhao H, Zhan J (2020) Named entity recognition only from word embeddings. EMNLP
32. Miech A, Zhukov D, Alayrac JB et al (2019) Howto100m: learning a text-video embedding by watching hundred million narrated video clips. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 2630–2640
33. Mikolov T, Sutskever I, Chen K et al (2013) Distributed representations of words and phrases and their compositionality. In: NeurIPS
34. Park J, Lee J, Sohn K (2021) Bridge to answer: structure-aware graph interaction network for video question answering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 15,526–15,535
35. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: EMNLP
36. Peters ME, Neumann M, Iyyer M et al (2018) Deep contextualized word representations. In: NAACL
37. Pfeiffer J, Vulić I, Gurevych I et al (2020) Mad-x: an adapter-based framework for multi-task cross-lingual transfer. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 7654–7673
38. Sanh V, Debut L, Chaumond J et al (2019) Distilbert, a distilled version of bert: smaller, faster cheaper and lighter. In: NeurIPS Workshop
39. Sennrich R, Haddow B, Birch A (2016) Neural machine translation of rare words with subword units. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1 : Long Papers), pp 1715–1725
40. Seok M, Song HJ, Park CY et al (2016) Named entity recognition using word embedding as a feature. Int J Softw Eng Appl 10(2):93–104
41. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
42. Soomro K, Zamir AR, Shah M (2012). In: Ucf101: a dataset of 101 human actions classes from videos in the wild. arXiv:12120402
43. Standley T, Zamir A, Chen D et al (2020) Which tasks should be learned together in multi-task learning? In: International conference on machine learning, PMLR, pp 9120–9132
44. Sun G, Liang L, Li T et al (2021) Video question answering: a survey of models and datasets. Mob Netw Appl, 1–34
45. Tapaswi M, Zhu Y, Stiefelhagen R et al (2016) Movieqa: understanding stories in movies through question-answering. In: CVPR
46. Tran D, Bourdev L, Fergus R et al (2015) Learning spatiotemporal features with 3d convolutional networks. In: ICCV

47. Ulmer D, Hardmeier C, Frellsen J (2022) Deep-significance-easy and meaningful statistical significance testing in the age of neural networks. arXiv:220406815
48. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. In: NeurIPS
49. Wang Y, Zhai C, Awadalla HH (2020) Multi-task learning for multilingual neural machine translation. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 1022–1034
50. Wang J, Bao B, Xu C (2021) Dualvgr: a dual-visual graph reasoning unit for video question answering. IEEE Trans Multimedia
51. Winterbottom T, Xiao S, McLean A et al (2020) On modality bias in the tvqa dataset. BMVC
52. Wolf T, Debut L, Sanh V, other (2019) Transformers: state-of-the-art natural language processing. arXiv:191003771
53. Wu Y, Schuster M, Chen Z, other (2016) Google's neural machine translation system: bridging the gap between human and machine translation. arXiv:160908144
54. Xiao J, Shang X, Yao A et al (2021) Next-qa: next phase of question-answering to explaining temporal actions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9777–9786
55. Xu D, Zhao Z, Xiao J et al (2017) Video question answering via gradually refined attention over appearance and motion. In: ACM Multimedia
56. Xu H, Ghosh G, Huang PY et al (2021) Vlm: task-agnostic video-language model pre-training for video understanding. In: Findings of the association for computational linguistics : ACL-IJCNLP, pp 4227-4239
57. Yang Z, Garcia N, Chu C et al (2020) Bert representations for video question answering. In: WACV
58. Yang A, Miech A, Sivic J et al (2021) Just ask: learning to answer questions from millions of narrated videos. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1686–1697
59. Zamir AR, Sax A, Cheerla N et al (2020) Robust learning through cross-task consistency. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11,197–11,206
60. Zhang Y, Yang Q (2021) A survey on multi-task learning. IEEE Trans Knowl Data Eng
61. Zhao Z, Zhang Z, Xiao S et al (2018) Open-ended long-form video question answering via adaptive hierarchical reinforced networks. In: IJCAI, p 4
62. Zhu L, Yang Y (2020) Actbert: learning global-local video-text representations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8746–8755