Text-Guided Object Detector for Multi-modal Video Question Answering

Ruoyue Shen, Nakamasa Inoue, Koichi Shinoda
Tokyo Institute of Technology
ruoyue@ks.c.titech.ac.jp, inoue@c.titech.ac.jp, shinoda@c.titech.ac.jp

Abstract

Video Question Answering (Video QA) is a task to answer a text-format question based on the understanding of linguistic semantics, visual information, and also linguistic-visual alignment in the video. In Video QA, an object detector pre-trained with large-scale datasets, such as Faster R-CNN, has been widely used to extract visual representations from video frames. However, it is not always able to precisely detect the objects needed to answer the question because of the domain gaps between the datasets for training the object detector and those for Video QA. In this paper, we propose a text-guided object detector (TGOD), which takes text question-answer pairs and video frames as inputs, detects the objects relevant to the given text, and thus provides intuitive visualization and interpretable results. Our experiments using the STAGE framework on the TVQA+ dataset show the effectiveness of our proposed detector. It achieves a 2.02 points improvement in accuracy of QA, 12.13 points improvement in object detection (mAP50), 1.1 points improvement in temporal location, and 2.52 points improvement in ASA over the STAGE original detector.

1. Introduction

The past decade has seen the rapid development of deep learning in many fields, including computer vision (CV) and natural language processing (NLP). The goal of CV is to build a machine that can extract meaningful information from images, videos, or other visual inputs. NLP, another major domain for deep learning, aims to understand and utilize human language. Question answering [1, 2, 3] is its important problem, in which a model is required to answer a natural language question by referring to a structured knowledge database or unstructured text documents.

Information usually appears in more than one modality in the real world. In order to make neural networks understand the world better, multi-modal learning, a task to make a model which can interpret and reason from different modalities, has attracted a lot of interest. As a derivative of question answering, a model in video question answering (Video QA) [4, 5, 6, 7] takes a video and the corresponding question as inputs and answers the question based on its understanding of linguistic and visual information. It can be used for many applications, such as video reviews, smart robots, and personal assistants.

Several Video QA studies have utilized deep learning recently. A standard Video QA system [8, 4, 9] consists of visual information extractor, textual information extractor, and multi-modal fusion module. For visual information in videos, there are three fine- to coarse-grained levels, i.e., region, frame, and clip level. The widely used approach is the region-level object feature extractor [10, 8, 4, 11], which is usually the Faster RCNN [12] that has a ResNet-101 backbone and is pre-trained on the Visual Genome dataset [13]. However, it is trained in an object detection task whose target and data are largely different from those of the Video QA task. Accordingly, the detection result is not precise enough for the subsequent fusion and prediction procedure, thus limiting QA performance.

To solve this problem, we propose the Text-Guided Ob-
Our contribution is two-fold: guided object detector on the Video QA task. In summary, to our best knowledge, we are the first to train a text-guided visual features for answering the related question. Some object detection methods with text guidance [14, 15, 16, 17] are proposed recently. STVGBert[17] is the most similar work to TGOD. Its highlighted ST-VILBERT module uses co-attention with two branches to extract text-guided visual features. However, these features are extracted by TGOD with the novel-designed multi-modal token sequences as the input to a single branch transformer decoder. This design saves computations and directly outputs regional object features for answering the related question. To our best knowledge, we are the first to train a text-guided object detector on the Video QA task. In summary, our contribution is two-fold:

1. We propose the Text-Guided Object Detector (TGOD) for Video Question Answering. TGOD utilizes the text features from Question-Answer pairs in the process of object detection to detect the objects relevant to the QA pairs.

2. We show that the proposed TGOD improved the interpretability and the performance in terms of several metrics. It is evaluated using STAGE [4] framework on the TVQA+ dataset. Compared with the original detector in STAGE, it improves the accuracy of QA by 2.02 points, object detection (mAP50) by 12.13 points, temporal location by 1.1 points, and ASA by 2.52 points. Its detection visualization also indicates interpretability improvement.

2. Related Work

2.1. Video Question Answering

The goal of Video QA is to answer the questions either in the form of free text in natural language or by selecting one answer out of a set of multiple candidate choices. Typically, a Video QA model consists of visual information extractor, textual information extractor, and multi-modal fusion module.

A video contains visual features of different levels, from fine-grained to coarse-grained. There are mainly three kinds of visual information extractors in Video QA models. Region-level feature extractors[8, 4, 11] are widely used and are usually Faster R-CNN [12]. Frame-level and clip-level feature extractors are usually convolutional neural networks (CNN)[9, 18, 19] and 3D CNN[20, 6, 21] respectively in early works, while recently vision transformers based extractors[22, 23, 24] are explored inspired by its promising results in various CV tasks[17, 25, 26, 27].

Linguistic information is highly abstract and can be well encoded using pre-trained language models. The early studies [8, 19, 9, 20] usually used Word2Vec and GloVe as the textual information extractor to extract static word embeddings, whereas recently, contextual word embeddings extracted by BERT [28, 4, 29, 7] are more preferred because of its excellent performance.

The multi-modal fusion module can be subdivided into three categories: encoder-decoder methods [6, 30, 8], memory network-based methods [7, 31, 29, 19, 9], and attention-based methods [4, 20, 21, 32, 11]. The commonly used design is attention-based ones, which pays more attention to the important part of the input. Transformer structure is also widely used [22, 23, 33, 24] as a novel kind of attention-based methods.

Among them, STAGE [4] uses Faster R-CNN to extract region-level visual features, BERT to extract text features, QA-guided attention to fuse features from the two modalities and then make the prediction. It is one of the most promising methods because it jointly performs spatial objects and temporal spans grounding with question answering. These spatial and temporal predictions help improve the performance of QA and also provide explainable results.

2.2. Object Detection

Object detection has long been an important and challenging task in the field of CV, requiring the separation of multiple objects from the background and the identification of their specific locations and categories.

R-CNN [34], a seminal work on using deep learning for object detection, divides images into multiple regions, feeds them into a CNN to extract region-level features, and uses a classifier and a regressor to get the prediction. Faster R-CNN [12] shares the convolution and uses the RPN network to extract ROI to do end-to-end detection, which also lays the foundation for two-stage models [35, 36], namely RPN+R-CNN. FPN [37] with lateral connections and top-down paths aims to handle objects of different sizes in the same semantic scene, and is widely used by following algorithms [38, 39, 40].

Two-stage models bring with them additional time and space overheads. One-stage models using unified end-to-end regression to obtain both object locations and categories simultaneously [38, 41, 40, 42, 43] comes on the stage. RetinaNet [40] uses the focal loss to alleviate the extreme imbalance between positive and negative samples in one-stage detectors and significantly improves the performance.

A large number of studies [25, 26, 44, 27, 16] transferring Transformer to CV have proven that the powerful representation capability brought by self-attention gives them
3. Text-Guided Object Detector

This section presents our proposed method, an object detector for Video QA, namely Text-Guided Object Detector (TGOD).

3.1. Overview

Figure 2 shows an overview of TGOD. It consists of three components: a) visual encoder, b) text encoder and c) object decoder. Unlike the traditional object detectors widely used in Video QA that detect all the items appearing in the image under the pre-defined categories [10, 4, 45, 11], TGOD only detects those objects relevant to QA pairs.

First, the visual encoder extracts visual feature maps from each input video frame by using a CNN backbone, and feeds them with the position and scale encodings into the transformer encoder to obtain the visual representation $V$. The input $Z$ to the transformer encoder is a transformation of the sum of flattened feature maps, scale embeddings, and 2D position embeddings. Specifically, a fixed position embedding $\{e_b\}_{b=2}^{B}$ and a trainable scale embedding $\{s_b\}_{b=2}^{B}$ is applied to each feature map as $z_b = f_b(v_b + e_b + s_b)$, where $f_b$ is a trainable linear layer that reduces the dimension from $C_h$ to a lower hidden dimension $D$. The position embeddings help to avoid the permutation-invariant problem, while scale embeddings identify the feature level each pixel lies in. Each $z_b \in \mathbb{R}^{D \times H_b \times W_b}$ is then reshaped to a matrix of shape $(D, L_b)$ by applying a flatten function to it, where $L_b = H_b W_b$. All the reshaped $z_b$ are concatenated into $Z \in \mathbb{R}^{D \times L}$ where $L = \sum_{b=2}^{B} L_b$. This $Z$ is a sequence of $D$-dimensional vectors of length $L$ and is used as the input sequence of the transformer encoder.

The transformer encoder is a stack of six encoder layers, each of which consists of a multiscale deformable self-attention (MSDeformAttn) [47] and a feed-forward network (FFN). In the self-attention, the $D$-dimensional query el-
element is each pixel in the feature map $Z$. The attention weights and sampling offsets are calculated for each query pixel by linear projections, and the reference point is the query pixel’s coordinates. The FFN consists of two linear layers with a ReLU activation in between. The final output is the visual representation $V$ that has the same dimensions as the input $Z$.

### 3.3. Text Encoder

The text encoder, composed of a BERT embedder and a POS tagging module, is used to encode QA pairs into text-feature representations $T$.

**BERT embedder.** The input QA pair is embedded by the BERT language model [48] as

$$S = \text{BERT}([Q, A]),$$  \hfill (1)

where $Q$ is the question sentence and $A$ is the answer sentence. Note that the question sentence and the answer sentence are concatenated into a single sequence of words $(w_1, w_2, \ldots, w_l)$. The shape of $S$ is given by $(d, l)$, where $d$ is the embedding dimension for each word and $l$ is the number of words in the input sentence. We use the embedding extracted from the second-to-last layer of BERT, whose dimension $d$ is 768. The BERT model is pre-trained on the large-scale BookCorpus dataset, fine-tuned on all text data of the TVQA+ dataset\(^1\), and then frozen during the training of TGOD.

**POS tagging.** In parts-of-speech (POS) tagging, we assign a part-of-speech label to each word in a sentence, and filter out words that are unlikely to be visual objects to let TGOD focus more on distinguishing between possible object words. The transformer-based pipeline of spaCy [49] pre-trained on numerous English texts is used as a POS tagger. Only the feature representations of two kinds of POS words are selected as the input to the object decoder: (1) PROPN: proper noun, e.g., Sheldon, Amy; (2) NOUN: noun, e.g., girl, lunch. More specifically, text feature representations $T$ are extracted in the following two steps given the input sequence of words $(w_1, w_2, \ldots, w_l)$ and embedding $S = (s_1, s_2, \ldots, s_l) \in \mathbb{R}^{d \times l}$. First, the POS tagger is applied to an input sentence to obtain a set of proper nouns and nouns $P = \{p_j\}_{j=1}^{l'}$, where $l'$ is the length of selected words. Second, for each $p_j$, its embedding is computed as follows:

$$t_j = f \left( \frac{1}{N} \sum_{i=1}^{l} \delta(w_i = p_j) s_i \right),$$  \hfill (2)

where $\delta(\cdot)$ is a function that returns one if the input equation is true and zero otherwise. $N = \sum_{i=1}^{l} \delta(w_i = p_j)$ is the number of appearances of the word $p_j$ in the input sentence, and $f$ is a trainable linear layer. We denote the same hidden dimension of $t_j$ by $d'$. The output text-feature representation is the sequence $T = (t_1, t_2, \ldots, t_{l'}) \in \mathbb{R}^{d' \times l'2}$.

### 3.4. Object Decoder

Our object decoder is a composition of a transformer decoder and prediction heads, takes the text-feature representation $T$ and the visual representation $V$ as inputs, and outputs pairs of a predicted bounding box and its corresponding word label.

**Decoder input.** As shown in Figure 2 (c), the input of the transformer decoder $X$ is a sequence of tokens, which contains three parts, object tokens $o_i$, the special token [TEXT], and word tokens $t_j$, as follows:

$$X = (o_1, o_2, \ldots, o_{l_o}, \text{[TEXT]}, t_1, t_2, \ldots, t_{l_w}),$$  \hfill (3)

where $l_o$ and $l_w$ are pre-defined numbers of the object tokens and word tokens, respectively. All object tokens $o_i$ are zero vectors of dimension $d'$, because trainable embeddings (object queries) are added to them at each decoder layer, as explained later. The special [TEXT] token is a trainable $d'$-dimensional vector. This is a buffer between different types of input. We only have $l_{sp} = 1$ special token. The word tokens are the text representation $t_1, t_2, \ldots, t_{l'}$ obtained from the text encoder (Eq. (2)). Here, we set $l_w$ such that it is larger than $l'$ for any QA pairs. Note that zero padding is used in order to have a fixed length sequence, i.e., the word tokens are given by $t_1, t_2, \ldots, t_{l'}, 0_{l'_o+1}, \ldots, 0_{l_w}$ where $0_j$ are zero vectors. The total length of the input sequence is given by $l_{total} = l_o + l_{sp} + l_w$.

**Transformer decoder.** The transformer decoder is a stack of six layers, each of which includes four sub-layers: a position embedding layer, a self-attention layer, a cross-attention layer, and an FFN. The position embedding layer adds two kinds of position embeddings: object queries [44] and the type encoding to the input. The sequence of object queries is given by

$$(q_1, q_2, \ldots, q_{l_o}, 0, 0, \ldots, 0)_{l_{sp}+l_w}$$  \hfill (4)

where $(q_i)_{i=1}^{l_o}$ is a set of trainable vectors, and 0 is a zero vector. The sequence of type encoding is given by

$$(0, 0, \ldots, 0, \text{t}^*, \text{t}^*, \ldots, \text{t}^*)_{l_{sp}+l_w}$$  \hfill (5)

where $\text{t}^*$ is another trainable vector. The object queries and type encoding are shared across all decoder layers to provide object location guidance and distinguish different kinds of input tokens, respectively.

\(^1\)For a fair comparison, we use the BERT features used in STAGE [4] as input instead of fine-tuning BERT again in our experiments.

\(^2\)The order of $t_j$ is the same as that in the input words.
The self-attention uses the standard multi-head attention, in which the input tokens interact with each other, while the cross-attention is implemented as a multiscale deformable attention, computing attentions between input tokens \( X \) and the visual representation \( V \). The FFN outputs \( d' \)-dimensional vectors, and thus the output of each decoder layer is a vector sequence of length \( l_{\text{total}} \). With the decoder layers, the first \( l_o \) output vectors, i.e., output object tokens, will gradually learn the content information.

**Prediction heads.** The prediction heads consist of two independent modules: a bounding-box predictor and a label predictor. Each module takes as input the output object tokens \( \hat{o}_1, \hat{o}_2, \ldots, \hat{o}_{\ell} \) obtained from the final decoder layer. The bounding-box predictor is a 3-layer perceptron that outputs the relative offset w.r.t. the reference points of \( \hat{o}_i \). The label predictor is a linear layer that outputs a \( l_w \)-dimensional vector \( \hat{q}_i \). The \( j \)-th element of \( \hat{q}_i \) corresponds to a confidence score of the word \( p_j \) obtained from the POS tagger. Note that the number of objects detected in a video frame is usually smaller than the pre-defined number of object tokens \( l_o \), and there are some tokens that don’t correspond to any object. Thus, the last dimension of \( \hat{q}_i \) represents a special class \( \varnothing \) (not matched), and the last word token is always assumed to be a pad token. This special class is used when the bounding box is not matched with any visual object.

### 3.5. Loss function

Unlike the standard object detector which predicts an object category for each bounding box given a pre-defined set of object categories, TGOD predicts a matching between a word and a bounding box. A combination of bounding box loss \( \mathcal{L}_o \), label cross-entropy loss, and contrastive feature loss \( \mathcal{L}_w \) is used to train TGOD.

**Bounding box loss.** The target for the bounding box predictor is a four-dimensional vector \( b \) representing the position and size of the bounding box. The loss is defined by

\[
\mathcal{L}_{\text{box}} = \sum_{i=1}^{\ell^\text{GT}} L_{\text{GIoU}}(b'_i, \hat{b}'_i) + \lambda_{\text{L1}} \|b_i - \hat{b}_i\|_1, \tag{6}
\]

where \( \hat{b}_i \in \mathbb{R}^4 \) is the bipartite-matched [44] bounding box prediction, \( b_i \) is its ground-truth, \( b'_i, \hat{b}'_i \) is the top left and bottom right coordinate format bounding box converted from \( b_i, \hat{b}_i \), \( \ell^\text{GT} \) is the number of ground-truth bounding boxes, \( \lambda_{\text{L1}} \) is a weight, and \( L_{\text{GIoU}} \) is the scale-invariant Generalized IoU loss [50] given by

\[
L_{\text{GIoU}}(b', \hat{b}') = 1 - \left( \frac{|b' \cap \hat{b}'|}{|b' \cup \hat{b}'|} - \frac{|C \setminus (b' \cup \hat{b}')|}{|C|} \right). \tag{7}
\]

Here, \( \cup \) and \( \cap \) indicate the overlap and union region between two bounding boxes, respectively, and \( C \) is the smallest convex hull that encloses both \( b' \) and \( \hat{b}' \).

**Label cross-entropy loss.** The target for the label predictor is a one-hot encoding \( y \) of dimension \( l_w \), whose \( j \)-th element \( (j < l_w) \) corresponds to the word \( p_j \), and the last dimension corresponds to \( \varnothing \). Given the output \( \hat{q}_i \in \mathbb{R}^{l_w} \) of the label predictor, the label cross-entropy loss is defined by

\[
\mathcal{L}_{\text{label}} = -\sum_{i=1}^{l_o} \sum_{j=1}^{l_w} w_j y_{ij} \log(\hat{q}_{ij}). \tag{8}
\]

where \( y_i \in \{0, 1\}^{l_w} \) is the ground-truth target, and \( w_j \) is the relative classification weight, which is 0.1 for the \( \varnothing \) class and 1.0 for the others. With this loss, TGOD learns to predict the most relevant word to each bounding box.

**Contrastive feature loss.** In order to enhance the information exchange between the two modalities, a bidirectional contrastive feature loss is used, which is defined by the sum of two losses: object contrastive loss \( \mathcal{L}_o \) and word contrastive loss \( \mathcal{L}_w \). The former is InfoNCE loss [51] over the output object tokens:

\[
\mathcal{L}_o = -\sum_{i=1}^{l_o} \log \left( \frac{\text{sim}(\hat{o}_i, t_j^\star)}{\sum_{k=1}^{l_w} \text{sim}(\hat{o}_i, t_k)} \right), \tag{9}
\]

where \( \hat{o}_i \) is the output object token, \( t_j \) is the word tokens of object decoder input \( X \), \( j^\star \) is the index of the word matching the \( i \)-th object, and \( \text{sim}(. , .) \) is the similarity between two kinds of tokens. The latter is similarly defined over word tokens:

\[
\mathcal{L}_w = -\sum_{j=1}^{l_w} \frac{1}{|I_j|} \sum_{i \in I_j} \log \left( \frac{\text{sim}(\hat{o}^\star_i, t_j)}{\sum_{k=1}^{l_o} \text{sim}(\hat{o}_k, t_j)} \right), \tag{10}
\]

where \( i^\star \) is the index of an object matching the \( j \)-th word, and \( I_j \) is the set of all the indexes matching the \( j \)-th word. The similarity is measured by \( \text{sim}(\hat{o}, t) = \exp \left( g_1(\hat{o})^T g_2(t) / \tau \right) \) where \( g_1 \) and \( g_2 \) are two linear layers that reduce the dimension to \( d_o \), and \( \tau \) is a hyperparameter.

**Total loss.** The total loss is a weighted sum of the three losses:

\[
\mathcal{L} = w_{\text{box}} \mathcal{L}_{\text{box}} + w_{\text{label}} \mathcal{L}_{\text{label}} + w_{\text{cl}} (\mathcal{L}_o + \mathcal{L}_w), \tag{11}
\]

where \( w_{\text{box}}, w_{\text{label}}, w_{\text{cl}} \) are weight hyperparameters.

### 4. Experiments

#### 4.1. Experimental Settings

**Dataset.** The TVQA+ Dataset [4] is used for evaluation. It contains 4,198 video clips, 29,383 multiple-choice QA
pairs, and 148,468 video frames with 310,826 bounding boxes. On average, there are 2,09 bounding box annotations for each image. 10.58 annotations for each question, and 2,527 object categories. The questions comprise two parts: a question part (what, who, where, etc.), and a temporal location part (before, when, after) to locate a small clip of the video indicating when things happened. For example, “What instrument is Raj playing when Raj and Howard have their show?”. For each question, there are five candidate answers, and one of them is the correct answer. This dataset is the first to provide both spatial and temporal annotation for the answers. It is challenging because it requires the model to locate the relevant temporal moment and recognize relevant visual concepts indicating the reason why it chooses the answer. We follow the official training, validation, and test-public splits to train and test the proposed method, which consists of 23,545, 3,017, and 2,821 questions, respectively. For pre-training, the COCO 2017 train set [52] is used, which consists of 118K images.

**Evaluation measures.** Video-QA performance is measured with the following four metrics [4]: 1) Classification accuracy (QA Acc), 2) Temporal mean Intersection-over-Union (T-mIoU), 3) Object grounding mean Average Precision (G-mAP), and 4) Answer-Span joint Accuracy (ASA).

In addition, object detection performance is measured with the following COCO metrics [52]: 1) AP (average precision over IoU thresholds from 0.50 to 0.95 with a step size 0.05), 2) AP\(\theta\) (AP at IoU = 0.01\(\theta\) for \(\theta\) = 50 and 75), 3) APs (AP for small objects with area \(h \cdot w < 32^2\)), 4) APm (AP for medium objects with area \(32^2 \leq h \cdot w < 96^2\)), 5) API (AP for large objects with area \(h \cdot w \geq 96^2\)), 6) ARP (average recall given \(p\) detection per image for \(p = 1, 10, 100\)).

### 4.3. Experimental Results

**Video QA performance.** We evaluate our proposed model TGOD with several previous studies on the TVQA+ dataset. These previous models are retrained on the TVQA+ dataset using the official code, if it was provided. The STAGE using Faster R-CNN as object detector [4] is chosen as our baseline. Table 1 shows the results on TVQA+ validation set. The TGOD STAGE outperforms all the previous models on all the metrics. It shows relative gains of 2.42 points in QA accuracy, 12.55 points in G-mAP, 1.33 points in T-mIoU, and 2.02 points in ASA compared with the original STAGE. This result shows that using TGOD as the object extractor indeed improves the Video QA performance and provides interpretable detection results. Table 2 shows the results on the TVQA+ test set. The TGOD STAGE model outperforms almost all the others except SSP [54] that uses a two-stage training strategy: self-supervised pretraining and auxiliary contrastive learning. Although SSP is better than TGOD in some metrics, it can’t perform object detection, thus, the result of TGOD is more interpretable. Compared with the vanilla STAGE, TGOD shows a 2.02 points improvement in QA Acc, 12.13 points improvement in G-mAP, 1.1 points improvement in T-mIoU, and 2.52 points improvement in ASA.

**Training and inference speed.** Table 3 reports the training and inference speed. Note that training uses batched input and only on frames with annotations, while inference uses batch size one and on all the frames (much more than annotated ones), so the inference speed is slower than training. As can be seen, the Video QA model and object detection speed of TGOD is 16% faster than that of the base-
Table 2. Comparison on TVQA+ public-test set. (%)

<table>
<thead>
<tr>
<th>Model</th>
<th>Box pred</th>
<th>QA Acc</th>
<th>G-mAP</th>
<th>T-mIoU</th>
<th>ASA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST-VQA[18]</td>
<td>-</td>
<td>48.28</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SSP [54]</td>
<td>-</td>
<td>76.21</td>
<td>-</td>
<td>39.03</td>
<td>31.05</td>
</tr>
<tr>
<td>Two-stream [8]</td>
<td>✓</td>
<td>68.13</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ISR [55]</td>
<td>✓</td>
<td>73.81</td>
<td>29.76</td>
<td>33.15</td>
<td>23.09</td>
</tr>
<tr>
<td>RHA [45]</td>
<td>✓</td>
<td>74.34</td>
<td>-</td>
<td>31.53</td>
<td>21.77</td>
</tr>
<tr>
<td>TGOD STAGE (Ours)</td>
<td>✓</td>
<td>74.51</td>
<td>41.28</td>
<td>31.75</td>
<td>22.44</td>
</tr>
<tr>
<td>Human [4]</td>
<td>-</td>
<td>90.46</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3. Computation time for training and inference (sec/video). TGOD STAGE (ours) is compared with the STAGE baseline using Faster R-CNN. Two A100 GPUs are used to measure the speed.

<table>
<thead>
<tr>
<th></th>
<th>OD</th>
<th>QA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Training</td>
<td>0.313</td>
<td>0.039</td>
<td>0.352</td>
</tr>
<tr>
<td>Inference</td>
<td>2.212</td>
<td>0.606</td>
<td>2.818</td>
</tr>
<tr>
<td>Ours Training</td>
<td>0.261</td>
<td>0.033</td>
<td>0.294</td>
</tr>
<tr>
<td>Inference</td>
<td>1.837</td>
<td>0.535</td>
<td>2.372</td>
</tr>
</tbody>
</table>

Table 4. COCO object detection metrics on TVQA+ validation set. (%)

<table>
<thead>
<tr>
<th>Model</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>APs</th>
<th>APlm</th>
<th>API</th>
<th>AR1</th>
<th>AR10</th>
<th>AR100</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAGE</td>
<td>9.3</td>
<td>25.4</td>
<td>4.6</td>
<td>0.6</td>
<td>4.1</td>
<td>13.4</td>
<td>11.1</td>
<td>15.8</td>
<td>15.8</td>
</tr>
<tr>
<td>TGOD STAGE (Ours)</td>
<td>22.0</td>
<td>37.8</td>
<td>21.1</td>
<td>1.1</td>
<td>8.4</td>
<td>29.7</td>
<td>24.0</td>
<td>28.1</td>
<td>28.2</td>
</tr>
</tbody>
</table>

Object detection performance. Table 4 lists the object detection performance in terms of the COCO metrics for the TGOD STAGE and vanilla STAGE. TGOD outperforms the Faster R-CNN used by the baseline on all metrics, revealing its high performance in the object detection task. Figure 3 illustrates the adaptability of TGOD. Each row shows the same visual frame input with different text inputs (QA pairs). In TGOD, the attended important objects change with the text input.

Analysis by VQ Types. Table 5 shows the performance for different question types on the TVQA+ validation set. TGOD has a great improvement for the questions related to visual objects (‘what’, ‘who’, ‘where’), and achieves the best performance, indicating its superiority. For ‘why’ questions, the performance of TGOD is lower than that of RHA, which uses additional object and label relationship information, but it still outperforms the STAGE baseline. Surprisingly, TGOD improves the accuracy of ‘how’ questions, even if they are not usually directly related to visual objects, revealing that the feature extracted by TGOD is more general and can be of help in the reasoning of the following process to some extent.

Figure 4 shows some examples of object detection results using the Faster R-CNN detector in STAGE baseline, TGOD, and the ground truth. It’s obvious that TGOD often detects the crucial objects to answer the question even if they are not annotated in the ground truth, indicating its robustness and interpretability. On the other hand, the Faster R-CNN tends to detect all the objects with a label irrelevant to the QA pairs. It has difficulty paying attention to essential items, and thus less interpretable and more easily predicts the wrong answer.

Ablation study. The ablation study was conducted on the TVQA+ valid set, as shown in Table 6. The first row is the performance of the complete TGOD STAGE, and we remove the POS tagging, contrastive loss supervision, and multiscale feature maps respectively from the row above it to measure its necessity. After removing the POS tagging, all the metrics dropped, especially QA Acc and G-
Table 5. QA Accuracy (%) by question type on TVQA+ validation set.

<table>
<thead>
<tr>
<th>Model</th>
<th>What (60.52%)</th>
<th>Who (10.24%)</th>
<th>Where (9.68%)</th>
<th>Why (9.55%)</th>
<th>How (9.05%)</th>
<th>Total (100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAGE [4]</td>
<td>70.70</td>
<td>71.52</td>
<td>70.55</td>
<td>76.74</td>
<td>68.50</td>
<td>71.10</td>
</tr>
<tr>
<td>RHA [45]</td>
<td>72.23</td>
<td>69.57</td>
<td>73.63</td>
<td>81.60</td>
<td>69.23</td>
<td>72.58</td>
</tr>
<tr>
<td>TGOD STAGE (Ours)</td>
<td>72.84</td>
<td>72.17</td>
<td>74.66</td>
<td>79.86</td>
<td>72.16</td>
<td>73.52</td>
</tr>
</tbody>
</table>

Figure 4. Examples of detected objects. (a) Our TGOD model can detect the key object toolbox to answer the question even if the ground truth didn’t include it, while the Faster RCNN in vanilla STAGE detects many dispensable objects that may harm performance. (b) Similarly, the Faster RCNN detects lots of objects, while our TGOD detects the key object, wine, correctly.

Figure 5. Ablation study. (%)

Table 6. Ablation study. (%)

<table>
<thead>
<tr>
<th>Model</th>
<th>QA Acc</th>
<th>G-mAP</th>
<th>T-mIoU</th>
<th>ASA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGOD STAGE</td>
<td>73.52</td>
<td>38.03</td>
<td>31.67</td>
<td>20.75</td>
</tr>
<tr>
<td>− POS tagging</td>
<td>72.93</td>
<td>34.79</td>
<td>31.55</td>
<td>20.56</td>
</tr>
<tr>
<td>− contrastive loss</td>
<td>72.18</td>
<td>34.28</td>
<td>31.01</td>
<td>19.72</td>
</tr>
<tr>
<td>− multiscale feat</td>
<td>70.86</td>
<td>29.05</td>
<td>30.49</td>
<td>19.25</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper, we proposed Text-Guided Object Detector (TGOD) for the Video Question Answering task. TGOD detects important objects appearing both in the video and in the question-answer pairs to improve the accuracy of object detection and performance of the Video QA task. Our experiments on the TVQA+ Dataset show that TGOD STAGE outperforms the original STAGE with Faster R-CNN detector by a large margin on all four metrics, and is competitive with previous works. This study therefore indicates that given more precise visual object features, the model can achieve stronger performance on the Video QA task.

Finally, we discuss limitations and future work. First, deeper information behind detected objects, like object relations, is not fully used, on which we observe that the main reason for false predictions lies. Future work should therefore include adding another branch to make use of these features. Second, this work was also limited to the rare number of Video QA datasets providing frame-level object annotation, making it less general. Future research should be undertaken to make a larger dataset with the QA-mentioned object bounding box and temporal annotations.

Acknowledgements This work was supported by JST CREST JPMJCR1687 and NEDO JPNP18002.
References


