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

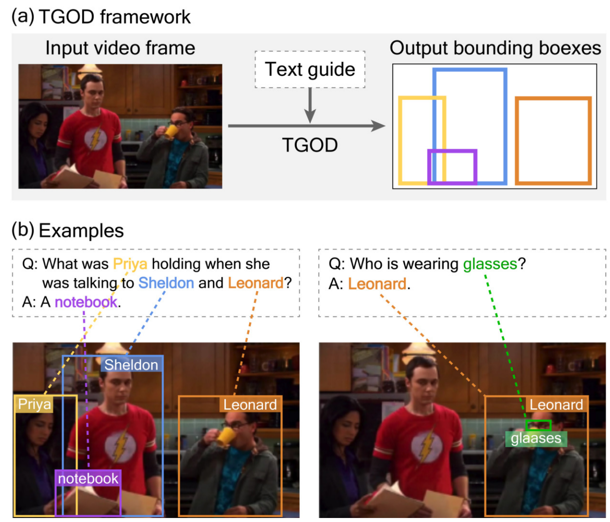

Figure 1. **Proposed text-guided object detector (TGOD).** (a) TGOD takes as input a video frame and text, and outputs bounding boxes for objects relevant to the text. (b) Two examples with different text guidance on the same video frame.

(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 Object Detector (TGOD), with which the model only pays attention to the objects relevant to the question-answer pairs and gets interpretable detection results. TGOD is a plug-in module and can be used to replace the object detector in Video QA models. As shown in Figure 1 (a), TGOD takes a video frame as input, uses question-answer pairs as text guiding signals, and outputs the bounding boxes of objects which appear in both modalities. Figure 1 (b) shows that TGOD's output differs according to the different text guidance. 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

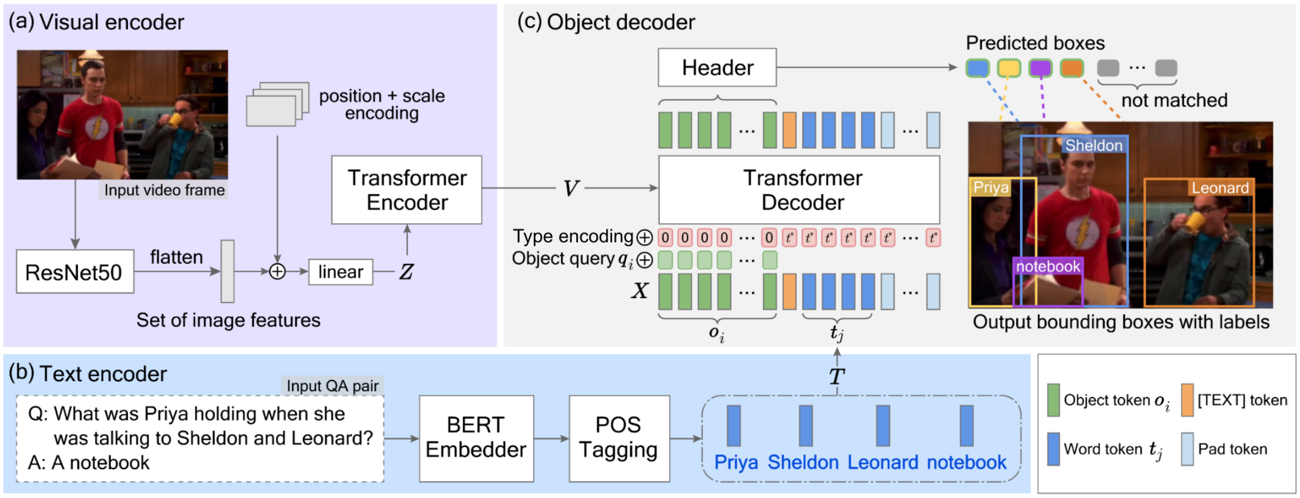

Figure 2. **Overview of proposed text-guided object detector (TGOD).** (a) The visual encoder takes a video frame as input, extracts multiscale feature maps by ResNet50, adds position and scale embeddings to provide spatial information, and flattens them into a sequence. Then, pass the flattened features to the transformer encoder to obtain a visual representation $\boldsymbol{V}$. (b) The text encoder uses a BERT embedder to encode the text, extracts nouns and proper nouns by POS tagging, and outputs their features as text representations $\boldsymbol{T}$. (c) The object decoder takes object tokens, a learnable [TEXT] token, and word tokens as inputs. Learnable object queries and type encoding are added at each transformer decoder layer. The prediction header takes output object tokens as inputs and predicts their bounding boxes and labels.

competitive or even superior performance on various vision tasks and benchmarks, including object detection.

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 $\boldsymbol{V}$. In parallel, the text encoder extracts word-level text feature representations $\boldsymbol{T}$ by using a pre-trained BERT embedder and selecting words that may correspond to an object among the input QA pair. Finally, the object decoder predicts bounding boxes with word labels based on the visual representation $\boldsymbol{V}$ and the text feature representation $\boldsymbol{T}$.

3.2. Visual Encoder

The visual encoder is composed of a CNN backbone and a transformer encoder. It takes a video frame as input and outputs its visual representation.

CNN backbone. The CNN backbone extracts multiscale visual feature maps from each input video frame. Assuming the CNN backbone consists of $B > 0$ convolutional blocks, the visual feature map $\boldsymbol{v}_b \in \mathbb{R}^{C_b \times H_b \times W_b}$ is extracted at each block and then transformed by a 1×1 convolution, where $b = 2, 3, \cdots, B$ is the index of the blocks and (C_b, H_b, W_b) is the channel, height, width of the feature map, respectively. With ResNet50 [46], we have $B = 5$ blocks with $C_b = 256$, $(H_b, W_b) = (H/2^b, W/2^b)$ where (H, W) is the size of the input video frame.

Transformer encoder. Given the feature maps $\{\boldsymbol{v}_b\}_{b=2}^{B}$, the transformer encoder produces a visual representation $\boldsymbol{V}$. The input $\boldsymbol{Z}$ 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 $\{\boldsymbol{e}_b\}_{b=2}^{B}$ and a trainable scale embedding $\{\boldsymbol{s}_b\}_{b=2}^{B}$ is applied to each feature map as $\boldsymbol{z}_b = f_b(\boldsymbol{v}_b + \boldsymbol{e}_b + \boldsymbol{s}_b)$, where f_b is a trainable linear layer that reduces the dimension from C_b 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 $\boldsymbol{z}_b \in \mathbb{R}^{D \times H_b \times W_b}$ 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 $\boldsymbol{z}_b$ are concatenated into $\boldsymbol{Z} \in \mathbb{R}^{D \times L}$ where $L = \sum_{b=2}^{B} L_b$. This $\boldsymbol{Z}$ 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 element is each pixel in the feature map $\boldsymbol{Z}$. 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 $\boldsymbol{V}$ that has the same dimensions as the input $\boldsymbol{Z}$.

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 $\boldsymbol{T}$.

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

$$\boldsymbol{S} = \mathrm{BERT}([Q, A]), \tag{1}$$

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 $(w_1, w_2, \cdots, w_l)$. The shape of $\boldsymbol{S}$ 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 $\boldsymbol{T}$ are extracted in the following two steps given the input sequence of words $(w_1, w_2, \cdots, w_l)$ and embedding $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_l) \in \mathbb{R}^{d \times l}$. 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:

$$\boldsymbol{t}_j = f\left(\frac{1}{N}\sum_{i=1}^{l}\delta(w_i = p_j)\boldsymbol{s}_i\right), \tag{2}$$

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 $\boldsymbol{t}_j$ by d'. The output text-feature representation is the sequence $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_{l'}) \in \mathbb{R}^{d' \times l'}$[2].

3.4. Object Decoder

Our object decoder is a composition of a transformer decoder and prediction heads, takes the text-feature representation $\boldsymbol{T}$ and the visual representation $\boldsymbol{V}$ 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 $\boldsymbol{X}$ is a sequence of tokens, which contains three parts, object tokens $\boldsymbol{o}_i$, the special token [TEXT], and word tokens $\boldsymbol{t}_j$, as follows:

$$\boldsymbol{X} = (\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_{l_\mathrm{o}}, [\mathrm{TEXT}], \boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_{l_\mathrm{w}}), \tag{3}$$

where l_o and l_w are pre-defined numbers of the object tokens and word tokens, respectively. All object tokens $\boldsymbol{o}_i$ 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 $l_\mathrm{sp} = 1$ special token. The word tokens are the text representation $\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_{l'}$ 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 $\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_{l'}, \boldsymbol{0}_{l'+1}, \cdots, \boldsymbol{0}_{l_\mathrm{w}}$ where $\boldsymbol{0}_j$ are zero vectors. The total length of the input sequence is given by $l_\mathrm{total} = l_\mathrm{o} + l_\mathrm{sp} + l_\mathrm{w}$.

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

$$(\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_{l_\mathrm{o}}, \underbrace{\boldsymbol{0}, \boldsymbol{0}, \cdots, \boldsymbol{0}}_{l_\mathrm{sp}+l_\mathrm{w}}) \tag{4}$$

where $\{\boldsymbol{q}_i\}_{i=1}^{l_\mathrm{o}}$ is a set of trainable vectors, and $\boldsymbol{0}$ is a zero vector. The sequence of type encoding is given by

$$(\underbrace{\boldsymbol{0}, \boldsymbol{0}, \cdots, \boldsymbol{0}}_{l_\mathrm{o}}, \underbrace{\boldsymbol{t}^*, \boldsymbol{t}^* . \cdots, \boldsymbol{t}^*}_{l_\mathrm{sp}+l_\mathrm{w}}) \tag{5}$$

where $\boldsymbol{t}^*$ 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 $\boldsymbol{t}_j$ 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 $\boldsymbol{X}$ and the visual representation $\boldsymbol{V}$. The FFN outputs d'-dimensional vectors, and thus the output of each decoder layer is a vector sequence of length l_{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 $\hat{\boldsymbol{o}}_1, \hat{\boldsymbol{o}}_2, \cdots, \hat{\boldsymbol{o}}_{l_{\text{o}}}$ 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 deformable attention, *i.e.*, a four-dimensional vector $\hat{\boldsymbol{b}}_i$ regarding the position and the size of a bounding box for each $\hat{\boldsymbol{o}}_i$. The label predictor is a linear layer that outputs a l_{w} dimensional vector $\hat{\boldsymbol{q}}_i$. The j-th element of $\hat{\boldsymbol{q}}_i$ 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 $\hat{\boldsymbol{q}}_i$ 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 [50], label cross-entropy loss, and contrastive feature loss [51] is used to train TGOD.

Bounding box loss. The target for the bounding box predictor is a four-dimensional vector $\boldsymbol{b}$ representing the position and size of the bounding box. The loss is defined by

$$\mathcal{L}_{\text{box}} = \sum_{i=1}^{l_{\text{o}}^{\text{GT}}} L_{\text{GIoU}}(\boldsymbol{b}'_i, \hat{\boldsymbol{b}}'_i) + \lambda_{\text{L1}} \|\boldsymbol{b}_i - \hat{\boldsymbol{b}}_i\|_1, \tag{6}$$

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

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

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 $\boldsymbol{b}'$ and $\hat{\boldsymbol{b}}'$.

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

$$\mathcal{L}_{\text{label}} = -\sum_{i=1}^{l_{\text{o}}} \sum_{j=1}^{l_{\text{w}}} w_j y_{ij} \log(\hat{q}_{ij}). \tag{8}$$

where $\boldsymbol{y}_i \in \{0,1\}^{l_{\text{w}}}$ 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 $\mathcal{L}_{\text{o}}$ and word contrastive loss $\mathcal{L}_{\text{w}}$. The former is InfoNCE loss [51] over the output object tokens:

$$\mathcal{L}_{\text{o}} = -\sum_{i=1}^{l_{\text{o}}} \log \left(\frac{\text{sim}(\hat{\boldsymbol{o}}_i, \boldsymbol{t}_{j^*})}{\sum_{k=1}^{l_{\text{w}}} \text{sim}(\hat{\boldsymbol{o}}_i, \boldsymbol{t}_k)} \right), \tag{9}$$

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

$$\mathcal{L}_{\text{w}} = -\sum_{j=1}^{l_{\text{w}}} \frac{1}{|I_j|} \sum_{i^* \in I_j} \log \left(\frac{\text{sim}(\hat{\boldsymbol{o}}_{i^*}, \boldsymbol{t}_j)}{\sum_{k=1}^{l_{\text{o}}} \text{sim}(\hat{\boldsymbol{o}}_k, \boldsymbol{t}_j)} \right), \tag{10}$$

where i^* 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 $\text{sim}(\hat{\boldsymbol{o}}, \boldsymbol{t}) = \exp\left(g_1(\hat{\boldsymbol{o}})^\top g_2(\boldsymbol{t})/\tau\right)$ where g_1 and g_2 are two linear layers that reduce the dimension to d_{o}, and τ is a hyperparameter.

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

$$\mathcal{L} = w_{\text{box}}\mathcal{L}_{\text{box}} + w_{\text{label}}\mathcal{L}_{\text{label}} + w_{\text{cl}}(\mathcal{L}_{\text{o}} + \mathcal{L}_{\text{w}}), \tag{11}$$

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) APθ (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) APl (AP for large objects with area $h \cdot w \geq 96^2$), 6) ARp (average recall given p detection per image for $p = 1, 10, 100$).

4.2. Implementation Details

Baseline Video QA architecture. The STAGE architecture [4] is chosen as the QA framework, which consists of an object detection module, text encoding module, and QA-attention module. The object detection module generates object representations from the video and is the module on which our proposed method TGOD is implemented. The text encoding module computes text embeddings for the hypothesis(QA pairs) and subtitle sentences with the BERT encoder [48]. All objects and text embeddings are inputted to the QA-attention module and prediction head to get the predicted answer. STAGE learns to answer questions via three losses (answer loss, span loss, and attention loss; see [4] for details). Note that we only modify the object detection module in this work.

TGOD architecture and hyperparameters. The visual encoder uses ResNet50 [46] as the CNN backbone. The hidden dimension of transformer encoder input D and that of decoder input d' are 256. Both the transformer encoder and decoder consist of six layers with eight attention heads and four reference points. The text encoder consists of the BERT model used in [4], and the POS tagger of spaCy [49]. The length of the object decoder input is 300 ($l_{\mathrm{o}} = 280$, $l_{\mathrm{sp}} = 1$, and $l_{\mathrm{w}} = 19$). The loss weights are $w_{\mathrm{box}} = 2, w_{\mathrm{label}} = 2, w_{\mathrm{cl}} = 1, \lambda_{\mathrm{L1}} = 2.5$. In contrastive feature loss, the dimension d_{o} is 300 and τ is 0.07.

Training details. The AdamW optimizer with a base learning rate of 2.0×10^{-4}, and weight decay of 10^{-4} is used for training for 50 epochs with a batch size of 16. The learning rate is decayed by a factor of 0.1 at epoch 30 and 40. The learning rates of the CNN backbone and of the object query linear projections are multiplied by a factor of 0.1. Two NVIDIA A100 GPUs are used in the experiments.

Table 1. Comparison on TVQA+ validation set (%).

Model	QA Acc	G-mAP	T-mIoU	ASA
ROLL [53]	69.61	-	-	-
STAGE [4]	71.10	25.48	30.34	18.73
RHA [45]	72.58	-	31.30	20.64
TGOD STAGE (Ours)	**73.52**	**38.03**	**31.67**	**20.75**

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. (%)

Model	Box pred	QA Acc	G-mAP	T-mIoU	ASA
ST-VQA[18]	-	48.28	-	-	-
SSP [54]	-	76.21	-	39.03	31.05
Two-stream [8]	✓	68.13	-	-	-
STAGE [4]	✓	72.49	29.15	30.65	19.92
ISR [55]	✓	73.81	29.76	33.15	23.09
RHA [45]	✓	74.34	-	31.53	21.77
TGOD STAGE (Ours)	✓	74.51	41.28	31.75	22.44
Human [4]	-	90.46	-	-	-

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.

		OD	QA	Total
Baseline	Training	0.313	0.039	0.352
	Inference	2.212	0.606	2.818
Ours	Training	0.261	0.033	0.294
	Inference	1.837	0.535	2.372

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

Model	AP	AP50	AP75	APs	APm	APl	AR1	AR10	AR100
STAGE	9.3	25.4	4.6	0.6	4.1	13.4	11.1	15.8	15.8
TGOD STAGE (Ours)	**22.0**	**37.8**	**21.1**	**1.1**	**8.4**	**29.7**	**24.0**	**28.1**	**28.2**

line using naïve Faster R-CNN. This is because TGOD has fewer parameters (41.0M vs. 60.6M), detects fewer objects and doesn't need a post-process procedure. It is also worth noting that the transformer encoder and decoder layers are composed of deformable attention [47], whose complexity is linear with the spatial size and is comparable with that of the convolution layers.

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 ro-

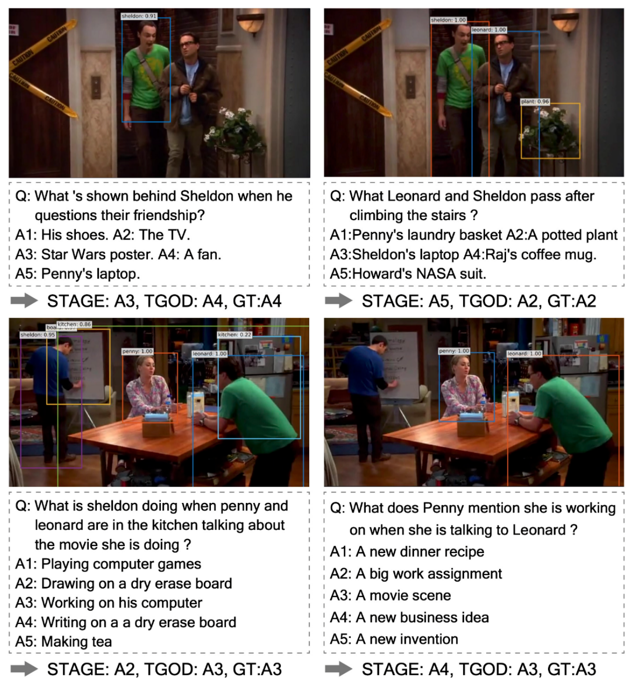

Figure 3. Examples of the adaptability of TGOD.

bustness 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.

Model	Question Type					
	What (60.52%)	Who (10.24%)	Where (9.68%)	Why (9.55%)	How (9.05%)	Total (100%)
STAGE [4]	70.70	71.52	70.55	76.74	68.50	71.10
RHA [45]	72.23	69.57	73.63	**81.60**	69.23	72.58
TGOD STAGE (Ours)	**72.84**	**72.17**	**74.66**	79.86	**72.16**	**73.52**

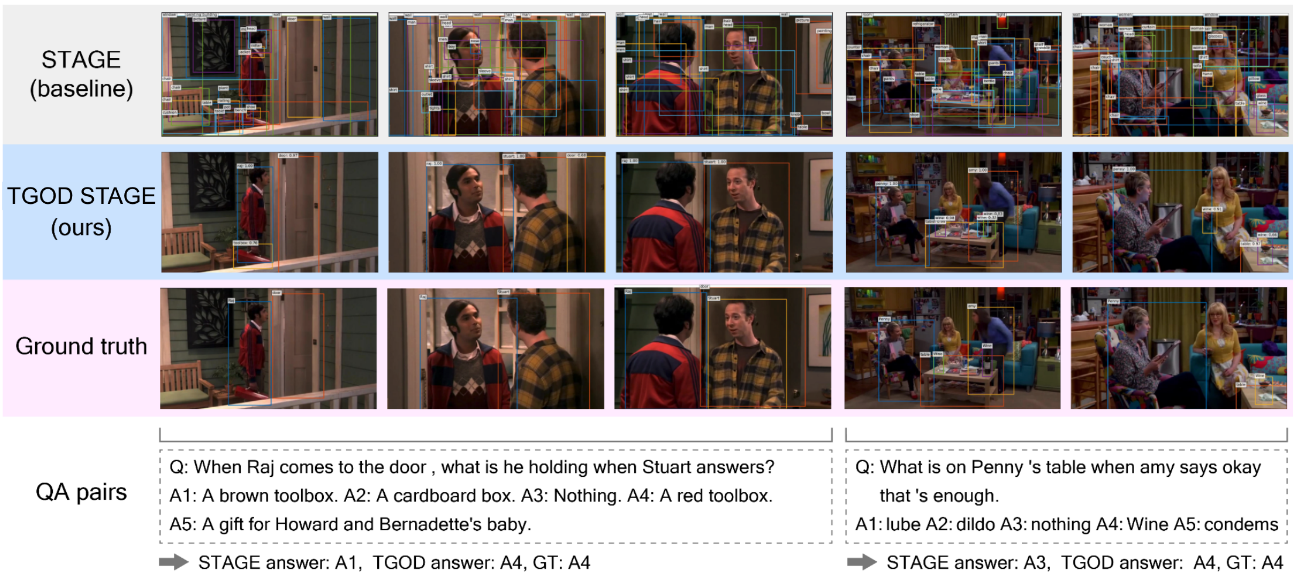

Figure 4. **Examples of detected objects.** (a) Our TGOD model can detect the key object toolbox to answer the question even if the ground truth didn't include it, while the Faster RCNN in vanilla STAGE detects many dispensable objects that may harm performance. (b) Similarly, the Faster RCNN detects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

mAP (0.59% and 3.24%), indicating negative word tokens filtration of POS tagging is crucial. The removal of contrastive loss supervision has less influence on object detection (0.51% drop for G-mAP) but has a significant influence on other metrics. It implies that the early alignment of visual and textual features in the object detection stage is helpful for the Video QA task. Lastly, using the last feature map instead of multiscale feature maps causes a significant performance drop for all four metrics (1.32%, 5.23%, 0.52%, 0.47% respectively). We believe the reason is that it's hard to use only the single-scale high-level feature to detect small objects, which is often the crucial clue to answer the question, leading to a huge performance drop.

Table 6. Ablation study. (%)

Model	QA Acc	G-mAP	T-mIoU	ASA
TGOD STAGE	73.52	38.03	31.67	20.75
– POS tagging	72.93	34.79	31.55	20.56
– contrastive loss	72.18	34.28	31.01	19.72
– multiscale feat	70.86	29.05	30.49	19.25

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

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[2] Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. Rikinet: Reading wikipedia pages for natural question answering. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[3] Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 8449–8456, 2020.

[4] Jie Lei, Licheng Yu, Tamara Berg, and Mohit Bansal. Tvqa+: Spatio-temporal grounding for video question answering. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8211–8225, 2020.

[5] Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L Berg, Mohit Bansal, and Jingjing Liu. Less is more: Clipbert for video-and-language learning via sparse sampling. In *Proc. International Conference on Computer Vision (CVPR)*, pages 7331–7341, 2021.

[6] Tegan Maharaj, Nicolas Ballas, Anna Rohrbach, Aaron Courville, and Christopher Pal. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. In *Proc. International Conference on Computer Vision (CVPR)*, pages 6884–6893, 2017.

[7] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. In *Proc. International Conference on Computer Vision (CVPR)*, pages 4631–4640, 2016.

[8] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara Berg. Tvqa: Localized, compositional video question answering. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1379, 2018.

[9] Junyeong Kim, Minuk Ma, Kyungsu Kim, Sungjin Kim, and Chang D Yoo. Progressive attention memory network for movie story question answering. In *Proc. International Conference on Computer Vision (CVPR)*, pages 8337–8346, 2019.

[10] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. International Conference on Computer Vision (CVPR)*, pages 6077–6086, 2018.

[11] Junyeong Kim, Minuk Ma, Kyungsu Kim, Sungjin Kim, and Chang D Yoo. Gaining extra supervision via multi-task learning for multi-modal video question answering. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

[13] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2016.

[14] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[15] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, et al. Regionclip: Region-based language-image pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16793–16803, 2022.

[16] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[17] Rui Su, Qian Yu, and Dong Xu. Stvgbert: A visual-linguistic transformer based framework for spatio-temporal video grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1533–1542, 2021.

[18] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proc. International Conference on Computer Vision (CVPR)*, pages 2758–2766, 2017.

[19] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. In *Proc. International Conference on Computer Vision (ICCV)*, pages 677–685, 2017.

[20] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *Proc. ACM international conference on Multimedia (ACMMM)*, pages 1645–1653, 2017.

[21] Zhou Zhao, Jinghao Lin, Xinghua Jiang, Deng Cai, Xiaofei He, and Yueting Zhuang. Video question answering via hierarchical dual-level attention network learning. In *Proc. ACM international conference on Multimedia (ACMMM)*, pages 1050–1058, 2017.

[22] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, 2019.

[23] Linchao Zhu and Yi Yang. Actbert: Learning global-local video-text representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8746–8755, 2020.

[24] Kevin Lin, Linjie Li, Chung-Ching Lin, Faisal Ahmed, Zhe Gan, Zicheng Liu, Yumao Lu, and Lijuan Wang. Swinbert: End-to-end transformers with sparse attention for video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17949–17958, 2022.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[26] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proc. International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021.

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. International Conference on Computer Vision (ICCV)*, 2021.

[28] Kyung-Min Kim, Min-Oh Heo, Seong-Ho Choi, and Byoung-Tak Zhang. Deepstory: video story qa by deep embedded memory networks. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2016–2022, 2017.

[29] Kuo-Hao Zeng, Tseng-Hung Chen, Ching-Yao Chuang, Yuan-Hong Liao, Juan Carlos Niebles, and Min Sun. Leveraging video descriptions to learn video question answering. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[30] Linchao Zhu, Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. Uncovering the temporal context for video question answering. *International Journal of Computer Vision*, 124(3):409–421, 2017.

[31] Jiyang Gao, Runzhou Ge, Kan Chen, and Ram Nevatia. Motion-appearance co-memory networks for video question answering. In *Proc. International Conference on Computer Vision (CVPR)*, pages 6576–6585, 2018.

[32] Kyung-Min Kim, Seong-Ho Choi, Jin-Hwa Kim, and Byoung-Tak Zhang. Multimodal dual attention memory for video story question answering. In *Proc. European Conference on Computer Vision (ECCV)*, pages 673–688, 2018.

[33] Gao Yuan Shi Yu Zhou Xiang-Dong Peng Min, Wang Chongyang. Multilevel hierarchical network with multiscale sampling for video question answering. *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[34] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. International Conference on Computer Vision (CVPR)*, pages 580–587, 2014.

[35] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 379–387, 2016.

[36] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[37] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. International Conference on Computer Vision (CVPR)*, pages 2117–2125, 2017.

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proc. European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[39] Peng Zhou, Bingbing Ni, Cong Geng, Jianguo Hu, and Yi Xu. Scale-transferrable object detection. In *Proc. International Conference on Computer Vision (CVPR)*, pages 528–537, 2018.

[40] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[41] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proc. International Conference on Computer Vision (CVPR)*, pages 7263–7271, 2017.

[42] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proc. International Conference on Computer Vision (CVPR)*, pages 4203–4212, 2018.

[43] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proc. European Conference on Computer Vision (ECCV)*, pages 385–400, 2018.

[44] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. European Conference on Computer Vision (ECCV)*, pages 213–229, 2020.

[45] Fangtao Li, Ting Bai, Chenyu Cao, Zihe Liu, Chenghao Yan, and Bin Wu. Relation-aware hierarchical attention framework for video question answering. In *Proc. International Conference on Multimedia Retrieval (ICMR)*, pages 164–172, 2021.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. International Conference on Computer Vision (CVPR)*, pages 770–778, 2016.

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[48] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[49] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[50] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proc. International Conference on Computer Vision (CVPR)*, pages 658–666, 2019.

[51] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[52] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[53] Noa Garcia and Yuta Nakashima. Knowledge-based video question answering with unsupervised scene descriptions. In *Proc. European Conference on Computer Vision (ECCV)*, pages 581–598, 2020.

[54] Seonhoon Kim, Seohyeong Jeong, Eunbyul Kim, Inho Kang, and Nojun Kwak. Self-supervised pre-training and contrastive representation learning for multiple-choice video qa. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 13171–13179, 2021.

[55] Rui Liu and Yahong Han. Instance-sequence reasoning for video question answering. *Frontiers of Computer Science*, 16(6):1–9, 2022.