STAIR: Spatial-Temporal Reasoning with Auditable Intermediate Results for Video Question Answering

Yueqian Wang1, Yuxuan Wang2,3, Kai Chen1, Dongyan Zhao1,3*
1Wangxuan Institute of Computer Technology, Peking University
2Beijing Institute for General Artificial Intelligence
3National Key Laboratory of General Artificial Intelligence
4School of Economics, Peking University
wangyueqian@pku.edu.cn, wangyuxuan1@bigai.ai, chen.kai@pku.edu.cn, zhaodongyan@pku.edu.cn

Abstract

Recently we have witnessed the rapid development of video question answering models. However, most models can only handle simple videos in terms of temporal reasoning, and their performance tends to drop when answering temporal-reasoning questions on long and informative videos. To tackle this problem we propose STAIR, a Spatial-Temporal Reasoning model with Intermediate Results for video question answering. STAIR is a neural module network, which contains a program generator to decompose a given question into a hierarchical combination of several sub-tasks, and a set of lightweight neural modules to complete each of these sub-tasks. Though neural module networks are already widely studied on image-text tasks, applying them to videos is a non-trivial task, as reasoning on videos requires different abilities. In this paper, we define a set of basic video-text sub-tasks for video question answering and design a set of lightweight neural modules to complete them. Different from most prior works, modules of STAIR return intermediate outputs specific to their intentions instead of always returning attention maps, which makes it easier to interpret and collaborate with pre-trained models. We also introduce intermediate supervision to make these intermediate outputs more accurate. We conduct extensive experiments on several video question answering datasets under various settings to show STAIR’s performance, explainability, compatibility with pre-trained models, and applicability when program annotations are not available. Code: https://github.com/yellow-binary-tree/STAIR

Introduction

Video question answering (video QA) is a challenging task that lies between the field of Natural Language Processing and Computer Vision, which requires a joint understanding of text and video to give correct answers. However, most approaches, including some recently proposed video-text large pre-trained models, only treat videos as animated images. They use black-box deep neural networks to learn mappings directly from inputs to outputs on factual questions like “Who is driving a car?”, ignoring the biggest difference between videos and images: the existence of temporal information. As a result, their performance tends to drop when understanding long and informative videos and answering complicated temporal-reasoning questions, such as determining the order of two events, or identifying events in a given time period of the video, where small differences in temporal expressions can lead to different results.

In comparison, in image question answering, many neural-symbolic methods have been proposed to tackle with complicated spatial-reasoning problems. Neural Symbolic VQA (Yi et al. 2018) aims to parse a symbolic scene representation out of an image, and converts the question to a program that executes on the symbolic scene representation. Neural Symbolic Concept Learners (Mao et al. 2019) also convert images to symbolic representations, but by learning vector representations for every visual concept. However, though these neural symbolic methods can achieve very good results on synthetic images like CLEVR (Johnson et al. 2016) and Minecraft (Wu, Tenenbaum, and Kohli 2017; Yi et al. 2018), they can not perform well on real-world images. One promising neural-symbolic approach is Neural Module Networks (NMNs) (Andreas et al. 2015). It first converts the question to a program composed of several functions using a program generator, and then executes the program by implementing each function with a neural network, which is also known as a “module”. With the introduction of neural networks at execution, it works better on real-word image question answering like VQA (Agrawal et al. 2015), and can also provide clues about its reasoning process by checking the program and inspecting the output of its modules.

In this paper we apply the idea of NMN to video question answering and propose STAIR, a Spatial-Temporal Reasoning model with Auditable Intermediate Results.

We define a set of basic video-text sub-tasks for video QA, such as localizing the time span of actions in the video, recognizing objects in a video clip, etc. We use a sequence-to-sequence program generator to decompose a question into its reasoning process, which is a hierarchical combination of several sub-tasks, and formalize this reasoning process into a formal-language program. Note that though the program generator requires question-program pairs to train, in practice we found that the program generator trained on AGQA2 (Grunde-McLaughlin, Krishna, and Agrawala 2022) question-program pairs (which is publicly available) can generate plausible programs for questions from other
datasets, so no further manual efforts are required to apply STAIR on video QA datasets without program annotations.

We also design a set of lightweight neural modules to complete each of these sub-tasks. These neural modules can be dynamically assembled into a neural module network according to the program. Then the neural module network takes video feature and text feature from a video encoder and a text encoder as input, and outputs a representation of the question after reasoning, which is then used by a classifier to generate the final answer. Different from most prior works of neural module networks, our neural modules return intermediate results specific to their intentions instead of always returning attention maps. Here we use the term “auditable” to describe that we can get the exact answer of each sub-task with no further actions required, which greatly increases the explainability of our method, and these intermediate results can also serve as prompts to improve the accuracy of pre-trained models. We also introduce intermediate supervision to make the intermediate results more accurate by training neural modules with ground truth intermediate results.

We conduct experiments on the AGQA dataset (Grunde-McLaughlin, Krishna, and Agrawala 2021, 2022), a large-scale, real-world video question answering dataset with most questions of it require combinational temporal and logical reasoning to answer, for a detailed analysis of STAIR. We also conduct experiments on STAR (Wu et al. 2021) and MSRVTT-QA (Gao et al. 2018) to test the feasibility of STAIR on datasets without human annotations of programs. In summary, the contributions of this paper include:

- We propose STAIR, a video question answering model based on neural module networks, which excels at solving questions that require combinational temporal and logical reasoning and is highly interpretable. We define sub-tasks for video QA, and design neural modules for the sub-tasks.
- We introduce intermediate supervision to make the intermediate results of the neural modules more accurate.
- We conduct extensive experiments on several video question answering tasks to demonstrate its performance, explainability, possibility to collaborate with pre-trained models, and applicability when program annotations are not available.

Related Works

Video Question Answering. Recent advances in video question answering methods can be roughly divided into four categories: (1) Attention based methods (Zhang et al. 2019; Li et al. 2019; Kumar et al. 2019) that adopt spatial and/or temporal attention to fuse information from question and video; (2) Memory network based methods (Xu et al. 2017; Gao et al. 2018; Fan et al. 2019; Kim et al. 2019) that use recurrent read and write operations to process video and question features; (3) Graph based methods (Jin et al. 2021; Söö et al. 2021; Xiao et al. 2021; Cherian et al. 2022; Park, Lee, and Sohn 2021; Zhao et al. 2022) that process videos as (usually object level) graphs and use graph neural networks to obtain informative video representations; and (4) Pre-trained models (Lei et al. 2021; Fu et al. 2021; Zellers et al. 2021, 2022; Wang et al. 2023) that pre-train a model in self-supervised manner with a mass of video-text multimodal data. Recently, many works also try to solve video QA in zero-shot settings using large pre-trained transformer-based models (Alayrac et al. 2022; Li et al. 2023; Zhang, Li, and Bing 2023; Lyu et al. 2023). Though many works have reported good video understanding and response generation abilities of their models, these models require massive computing resources to pre-train, and their training videos/questions are relatively simple in terms of temporal reasoning, which means that these models are not robust at understanding and reasoning temporal information of videos.

Since there is usually redundant information in the video, Some works (Kim et al. 2020; Gao et al. 2022; Li et al. 2022) also study helping the model focus on key information by selecting video clips relevant to the question.

Though the above-mentioned methods have achieved outstanding performance, for most of these methods their performance tends to drop when evaluating on questions that require complicated logical reasoning or counterfactual questions and are difficult to interpret. To tackle these problems, some works use neural symbolic approach (Yi et al. 2019) (Qian et al. 2022) or construct physics models (Ding et al. 2021; Chen et al. 2021).

Neural Module Networks. Neural Module Networks (NMN) have been widely used in image question answering (Andreas et al. 2015; Hu et al. 2017; Johnson et al. 2017; Mascharka et al. 2018; Hu et al. 2018). These methods explicitly break down questions into several sub-tasks and solve each of them with a specifically-designed neural network (module). Attention maps or image representations are used to pass information among modules. Neural Module Networks are generally more interpretable, and excel at tasks that require compositional spatial reasoning such as SHAPES (Andreas et al. 2015) and CLEVR (Johnson et al. 2016). A more advanced NMN for image-text tasks is the recently-proposed Visual Programming (Gupta and Kembhavi 2023). Taking advantage of several off-the-shelf models such as CLIP (Radford et al. 2021), GPT-3 (Brown et al. 2020) and Stable Diffusion (Rombach et al. 2021), Visual Programming is capable of performing image QA, object tagging, and natural language image editing without further training.

Contrary to the intense research efforts of NMNs on image QA, there are significantly fewer works that focus on video QA (Le, Chen, and Hoi 2022; Qian et al. 2022). Though sharing the same motivation, it is non-trivial to define the sub-tasks and design their corresponding modules for video modality, which is one of the main contributions of our work. The work most similar to ours is DSTN (Qian et al. 2022), which also uses neural module network for video QA. But our work is significantly different from theirs in better performance, better explainability, the usage of intermediate supervision, the ability to collaborate with pre-trained models, and verifying its applicability when program annotations are not available.
Methodology

In this section, we describe the details of STAIR. STAIR takes as input a video feature $x_v \in \mathbb{R}^{T \times \text{hid}_V}$ with $T$ frames encoded by a pre-trained visual feature extractor and a question $x_q$ with $L$ words, and selects an answer $a$ from a fixed set of all possible answers $A$. STAIR consists of the following components: (1) a bi-directional LSTM video encoder $ENC_{vid}$ which models the temporal relationship of the video feature and transforms it into the common hidden space $v = ENC_{vid}(x_v), v \in \mathbb{R}^{T \times H}$; (2) a bi-directional LSTM text encoder $ENC_{txt}$ which extracts the sentence-level and token-level question feature as $(q, t) = ENC_{txt}(x_q), q \in \mathbb{R}^H, t \in \mathbb{R}^{L \times H}$; (3) a collection of neural modules $\{f_m\}$, each of which has a set of associated parameters $\theta_m$, performs a specific sub-task, and can be combined into a neural module network; and (4) a two-layer classifier $\phi(\cdot)$ that predicts the final answer. Besides, a program generator $p = \text{gen}(x_q)$ is trained individually to predict the program that determines the layout of the modules given a question $x_q$. The overview of the model is shown in Figure 1.

Neural Modules

As mentioned above, our solving process of the questions can be decomposed into several sub-tasks. For example, to answer the question "After cooking some food what did they watch?", there are 3 sub-tasks to solve: first localize the clips among the entire video when the people are cooking, then navigate to clips that happen after the cooking clips, and finally focus on these clips to find out the object that the people are watching.

Our STAIR contains 16 neural modules implementing different sub-tasks. All of these modules are implemented by simple neural networks such as several linear layers or convolutional layers. Their inputs, outputs, and intended functions are very diverse, including Filter module that finds objects or actions from a given video clip, Exists module that determines whether an object exists in the results of Filter, Localize module that finds in which frames an action happens, to name a few. The intentions and implementation details of all modules are listed in the Appendix.

Different from most of the previous works of neural module networks, the inputs and outputs of our modules are not always the same (e.g., attention maps on images/videos), but are determined by the intentions of each module. Take the module $\text{Filter}(\text{video, objects})$ as an example, it intends to find all objects that appear in the video. Instead of returning an attention map showing when the objects occur, in our implementation it must return a feature vector from which we can predict the names of all objects in the video. This design leads to significantly better explainability and reliability, as we can know the exact objects it returns by only inspecting the output.

Programs and the Program Generator

The design of the program is inspired by the AGQA dataset. In AGQA, each question is labeled with a program con-
sisting of nested functions indicating the sub-tasks of this question, and each video is tagged with a video scene graph from Charades and Action Genome (Sigurdsson et al. 2016; Ji et al. 2019). The answer can be acquired by executing the program on the scene graph. We use a rule-based approach to convert the labeled program to a sequence of program tokens, which is the Reverse Polish Notation of the tree-structured layout of our modules. 1 To avoid confusion hereafter we refer to the program before and after conversion as sg_program and nnm_program. Note that though nnm_program is designed according to sg_program in AGQA, it also works on other video question answering tasks as shown in Section.

Program tokens in nnm_program can be categorized into 4 types: (1) module tokens which corresponds to a neural module, e.g., Filter, Localize; (2) the “video” token that represents the video feature v; (3) text tokens which corresponds to a text span in the question \( q \), e.g., “watch”, “cooking some food” and (4) switch tokens which are keywords switches between the branches in a module, e.g., “max”, “after”, “fwd”(“forward”).

As nnm_programs are not provided during inference, we need to train a program generator to learn the mappings from questions to nnm_programs. We tried fine-tuning a FLAN-T5-large (Wei et al. 2022), but this problem is easy as a simple bi-directional LSTM encoder-decoder model with attention can predict exactly the right nnm_program for more than 98% of the questions in AGQA, so we decide to use the light-weight LSTM here.

Intermediate Supervision

Previous works mentioned that sometimes modules in the neural module networks do not behave as we expected and thus can’t provide meaningful intermediate outputs for us to understand its reasoning steps despite predicting the final answer correctly (Hu et al. 2017). To mitigate this problem, we use intermediate supervision to induce supervision to intermediate modules. An example of intermediate supervision is shown in Figure 2. Given that nnm_program is obtained by converting sg_program using a rule-based approach, we can record the correspondence between functions in sg_program and modules in nnm_program.

![Figure 2: A Diagram of Intermediate Supervision.](image)

Then we execute sg_program on the video scene graph and take the return value of functions as ground truth answers of corresponding modules. 2 We use intermediate supervision for all but the first module in nnm_program (i.e., the root module in the tree structure), as the first module is already directly supervised by the answer. Note that intermediate supervision does not always improve the model’s performance, as its main purpose is to make the outputs of intermediate modules more accurate. Depending on the data type, we use different criteria to calculate the intermediate supervision loss \( L_{IS} \) between the gold answer and module prediction, which is elaborated in the Appendix.

Training Procedures

The program generator is trained individually, and the main model, including video encoder, text encoder, neural modules, and classifier are trained in an end-to-end manner.

Generating nnm_program is considered as a sequence-to-sequence task. A model \( gen(\cdot) \) takes question \( x_q \) as input and generate nnm_program \( \hat{p} \) in an auto-regressive manner:

\[
\log P(\hat{p}|x_q) = \sum_t \log(\hat{p}_t|\hat{p}_{<t})
\]

and the loss \( L^{GEN} \) is calculated using the negative log likelihood of ground truth nnm_program \( \hat{p} \):

\[
L^{GEN} = - \sum_t \log(p_t|x_q, p_{<t})
\]

When training the main model, the ground truth nnm_program of train and valid set, or the nnm_program generated by the program generator of test set is used to assemble the neural modules \( f_m \) into a tree-structured neural module network. The classifier loss \( L^{CLS} \) is calculated using the ground truth answer \( a \) and the predicted logits \( \hat{a} \) over all candidate answers produced by the classifier as:

\[
L^{CLS} = l_{ce}(\hat{a}, a)
\]

The total loss of the main model is \( L = L^{CLS} + \eta L^{IS} \), where \( \eta \) is a hyper-parameter balancing the classifier loss and the intermediate supervision loss.

Experiments

We evaluate STAIR mainly on AGQA balanced dataset (Grunde-McLaughlin, Krishna, and Agrawala 2021), as it is a large-scale, real-world video QA dataset with most questions in it requiring comprehensive temporal reasoning to answer. AGQA balanced dataset contains 3.9M question-answer pairs with 9.6K videos. Each question is associated

---

1For details of this rule-based approach please refer to our code.

2As the authors of AGQA and Action Genome do not release their code of acquiring answers via scene graphs, we have to implement these functions by ourselves. For about 5% of all training examples, our implementation can’t return the correct final answer given sg_program and scene graph of the corresponding video, so we don’t use intermediate supervision on them.
with a program that describes the reasoning steps to answer the questions. Videos in AGQA are from Charades (Sigurdsson et al. 2016), a diverse human action recognition dataset collected by hundreds of people in their own homes. Each video is annotated with a video scene graph containing spatial and temporal information about actions and objects from Action Genome (Ji et al. 2019). AGQA is very challenging, as even state-of-the-art deep learning models perform much worse than humans. We also evaluate on AGQA2 balanced dataset (Grunde-McLaughlin, Krishna, and Agrawala 2022) which contains 2.27M question-answer pairs selected with a stricter balancing procedure and is even more challenging than AGQA. Following (Le et al. 2020), we leave 10% of the train set out as valid set, and require videos in train/valid set to be different.

**Model Implementations**

**Implementation.** We used two different video features in our experiments. One is the standard video feature provided by the AGQA dataset, including appearance features $x^a_v \in R^{8 \times 16 \times 2048}$ extracted from ResNet-101 pool5 layer (He et al. 2015), and motion features $x^m_v \in R^{8 \times 2048}$ extracted from ResNetXt-101 (Xie et al. 2016). We use mean pooling on the second dimension of $x^a_v$ and concatenate it with $x^m_v$ to obtain the final video feature $x_v \in R^{8 \times 4096}$. We name this video feature “RX”. However, as the official RX feature only has 8 frames on temporal dimension which is insufficient for complicated temporal reasoning, we also extract a video feature ourselves. We sample frames from videos with a frame rate of 24 fps, and use an I3D model pretrained on Kinetics (Carreira and Zisserman 2017) to extract a 1024-d feature for every consecutive 16 frames. We clip the temporal dimension length to 64, so the final video feature is $x_v \in R^{1024 \times 1024}, T \leq 64$. We name this video feature “I3D”.

STAIR is trained with batch size 32, initial learning rate 2e-4 and decays linearly to 2e-5 in 200k steps. $\eta$ is set as 1. STAIR is trained on a Nvidia A100 GPU, and it takes about 2 epochs (30 hours) on average for a single run.

**Baselines.** We compare STAIR with and without intermediate supervision (-IS) with several baselines. We compare with 3 representative video QA models: HME (Fan et al. 2019) is a memory-network-based model to encode video and text features; HCRN (Le et al. 2020) uses conditional relational networks to build a hierarchical structure that learns video representation on both clip level and video level; PSAC (Li et al. 2019) uses both video and question positional self-attention instead of RNNs to model dependencies of questions and temporal relationships of videos. To compare with models that explicitly model the multi-step reasoning process, we also compare with DSTN (Qian et al. 2022), a neural module network concurrent to our work, and MAC (Hudson and Manning 2018) which performs iterative attention-based reasoning with a recurrent “Memory, Attention and Composition” cell. We make minor modifications on the attention of MAC to attend to 2-D ($T \times dim_v$) temporal features instead of 3-D ($H \times W \times dim_v$) spatial features.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Video</th>
<th>Binary</th>
<th>Open</th>
<th>Overall</th>
<th>#Prm</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSAC †</td>
<td>RX</td>
<td>53.56</td>
<td>32.19</td>
<td>42.44</td>
<td>39M</td>
</tr>
<tr>
<td>HME †</td>
<td>RX</td>
<td>57.21</td>
<td>36.57</td>
<td>46.47</td>
<td>42M</td>
</tr>
<tr>
<td>HCRN †</td>
<td>RX</td>
<td>56.01</td>
<td>40.27</td>
<td>47.82</td>
<td>41M</td>
</tr>
<tr>
<td>MAC</td>
<td>RX</td>
<td>57.74</td>
<td>41.24</td>
<td>49.15</td>
<td>16M</td>
</tr>
<tr>
<td>DSTN-E2E †</td>
<td>RX</td>
<td>57.38</td>
<td>42.43</td>
<td>49.60</td>
<td>36M</td>
</tr>
<tr>
<td>STAIR</td>
<td>RX</td>
<td>59.07</td>
<td>43.08</td>
<td>50.75</td>
<td>21M</td>
</tr>
<tr>
<td>STAIR-IS</td>
<td>RX</td>
<td>60.15</td>
<td>42.84</td>
<td>51.14</td>
<td>21M</td>
</tr>
<tr>
<td>MAC</td>
<td>I3D</td>
<td>58.19</td>
<td>46.84</td>
<td>52.28</td>
<td>10M</td>
</tr>
<tr>
<td>STAIR</td>
<td>I3D</td>
<td>60.18</td>
<td>47.24</td>
<td>53.45</td>
<td>14M</td>
</tr>
<tr>
<td>STAIR-IS</td>
<td>I3D</td>
<td>62.37</td>
<td>48.32</td>
<td>55.06</td>
<td>15M</td>
</tr>
</tbody>
</table>

Table 1: Results of AGQA. †: Results from (Qian et al. 2022). #Prm denotes number of parameters. #Prm of MAC varies slightly with its number of steps, here we show #Prm of a 12-step model.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Video</th>
<th>Binary</th>
<th>Open</th>
<th>Overall</th>
<th>#Prm</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>I3D</td>
<td>54.72</td>
<td>44.96</td>
<td>49.67</td>
<td></td>
</tr>
<tr>
<td>STAIR</td>
<td>I3D</td>
<td>57.13</td>
<td>47.07</td>
<td>52.06</td>
<td></td>
</tr>
<tr>
<td>STAIR-IS</td>
<td>I3D</td>
<td>56.48</td>
<td>46.41</td>
<td>51.41</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Results of AGQA2.

**Model Performance**

Table 1 shows the accuracy of all models on binary, open-ended and all questions of AGQA. STAIR outperforms all other baselines when using the same video feature, demonstrating the effectiveness of our approach. All models using the I3D video feature outperform their counterparts that use the RX feature, which shows the higher quality of I3D features. We also find that intermediate supervision does not always improve the performance of STAIR, probably due to the coordination problems among the losses of multi-task learning. However intermediate supervision does improve the model’s explainability by making the output of intermediate results more accurate, which is shown in the next subsection. We also compare STAIR with the strongest baseline MAC using the I3D video feature on AGQA2, and the results are shown in Table 2.

**Evaluation and Visualization of Modules’ Intermediate Output**

As our STAIR is based on neural module networks, it enjoys good interpretability while performing well. To demonstrate the interpretability of STAIR, we evaluate the intermediate results of Filter, Localize and Temporal modules, as these modules occurs at high frequency, and the outputs of them are intuitive and easy to inspect.

Filter module is designed to find objects and actions in the video or related to a given verb. To check the correctness of the output from Filter module, we use Recall@N in a retrieval task as the evaluation metric. We calculate a candidate representation for each of the 214 candidate answers, and use cosine similarity between the output of Filter module and candidate representations to select a list of N most likely predictions. If one of the predicted items occurs in the list of ground truth action(s)/object(s), we count it as a successful retrieval. We use the most frequently occurring
N actions/objects as baseline results.

Localize module is designed to find when an action happens in a video. We use \( \text{IoU}_{\text{att}} \) as the evaluation metric. Given the predicted and ground truth attention scores \( \text{att}_p, \text{att}_g \in R^T \), the metric \( \text{IoU}_{\text{att}} \) is calculated as

\[
\text{IoU}_{\text{att}} = \frac{\text{sum}(\min(\text{att}_p, \text{att}_g))}{\text{sum}(\max(\text{att}_p, \text{att}_g))},
\]

where \( \max \) and \( \min \) are element-wise operations. We use uniform distribution as baseline attention scores: \( \text{att}_b \sim \mathcal{U}(0, 1) \times T \).

Temporal module is designed to transform the attention scores according to the switch keyword \( s \). We use the same metric \( \text{IoU}_{\text{att}} \) to evaluate the attention scores output \( \text{att}_{\text{out}} \). Inspired by (Qian et al. 2022), we randomly sample two frames as the start and end frames as baseline results. Specially, the start frame is always the first frame when \( s = \text{‘before’} \), and the end frame is always the last frame when \( s = \text{‘after’} \).

Table 3 shows the results. STAIR performs baseline on most metrics except R@5 of Filter module, which indicates that STAIR is capable of providing meaningful intermediate results, and training with intermediate supervision can make the intermediate results more accurate.

We also visualize the reasoning process of STAIR on some real examples in the test set in the Appendix.

**Compatibility with Pre-trained Models**

Pre-trained models, including text-only ones and multimodal ones, have achieved state-of-the-art performance on many question answering tasks. Here we first compare STAIR with a single-modal pre-trained model GPT-2 (Radford et al. 2019), and a video-text pre-trained model Violet (Fu et al. 2021). For GPT-2, we prepend 13D video features to questions and assign different token type embeddings following (Li et al. 2020). For Violet, we sampled 10 video frames, resize them into 224 \( \times \) 224, and split them into patches with \( W \times H = 32 \times 32 \). Though we can’t use the pre-trained temporal position embedding as our \( T = 10 \) is larger than \( T = 4 \) in the pre-training stage and \( T = 5 \) for downstream tasks in the original paper, we find that this gives better results. Table 4 shows that on AGQA STAIR still underperforms GPT-2 and Violet, probably due to significantly fewer parameters and the absence of pre-training. However, the performance gap between STAIR and the pre-trained models on AGQA2 is smaller, probably due to the language bias being further reduced and it’s harder for pre-trained models to find textual clues to solve the questions.

To combine STAIR with pre-trained models, we use a straightforward method: we modify the questions to add the intermediate results of our neural modules to the input of pre-trained models as prompts. We get the top 1 candidate result for every Filter module in STAIR-IS using methods described in intermediate output subsection, and concatenate it with its keyword inputs. As Filter modules with lower levels have higher accuracy, we sort all Filter modules in ascending order of level and take only the first P modules into account, where P is selected in \( \{1, 3, 5\} \) by valid set performance. Take the following question as an example: *What did they take while sitting in the thing they went above?*. To answer this question, the corresponding nnn_program contains one Filter module with parameter (video, above) and returns the result “bag”. So the modified question becomes: *above bag. What did they take while sitting in the thing they went above?*. This can reduce the difficulty of questions by providing answers to some sub-tasks so it requires fewer steps to answer them. We use this method on the best-performing GPT-2 and denote it as GPT-2+STAIR-IS. Experiments show that with the help of these intermediate outputs, the performance of GPT-2 is further improved. It is also an evidence of the usefulness of the intermediate results.

Given the recent rapid development of multi-modal large pre-trained models, we also report the results of zero-shot Video-ChatGPT (Maaz et al. 2023), a video-text pre-trained model which is claimed to be optimized for temporal understanding in videos, and Video-ChatGPT + STAIR-IS in Table 5. Following (Maaz et al. 2023), Video-ChatGPT is not fine-tuned, and we benchmark its performance on AGQA2 with the evaluation pipeline using GPT-3.5. As it is unfeasible to test on the entire test set of AGQA2 with 660K questions, we randomly sample 1% (6.6K questions), repeat the experiment for 3 times, and report the average accuracy and standard deviation.

**Experiments on Tasks Without Program Annotations**

One may question that the need for program annotations limits the usage of STAIR. However, this question can be resolved by verifying that program generators trained on AGQA can be used to generate programs for questions from other video QA datasets: since the program annotations of AGQA is already publicly available, no more manual efforts are required to apply STAIR on datasets without program annotations.

To resolve this question, we conduct experiments on STAR (Wu et al. 2021) and MSRVTT-QA (Xu et al. 2017). We changed the program generator from an LSTM to a FLAN-T5-large (Wei et al. 2022) fine-tuned on AGQA2 question-nnn_program pairs to make the program generator more generalizable. Please refer to the Appendix for details of the experiments. Surprisingly, though the program generator has never seen questions from STAR and MSRVTT-QA during training phase, it can generate executable programs for more than 95% of the questions. Results are shown in Table 6 and Table 7. Though STAIR do not perform well on interaction type of questions as they are too simple to take advantage of the compositional ability of the neural modules, it outperforms several video question answering baselines on Sequence, Prediction and Feasibility.
Methods & AGQA & AGQA2 & #Params
--- | --- | --- | ---
STAIR & 60.18 & 47.24 & 53.45 & 14.97M
STAIR-IS & 62.37 & 52.88 & 57.13 & 15.11M
GPT-2 & 63.94 & 50.88 & 57.14 & 127M
Violet & 60.87 & 52.88 & 56.72 & 160M
GPT-2+STAIR-IS & 64.26 & 50.97 & 57.34 & 127M

Table 4: Results of AGQA and AGQA2, comparing with pre-trained models.

Methods  | Overall
--- | ---
Video-ChatGPT & 35.09 (0.76)
+ STAIR-IS & 40.43 (0.89)

Table 5: Results of Video-ChatGPT on AGQA2.

Methods & Int. & Seq. & Pre. & Fea. & Overall
--- | --- | --- | --- | --- | ---
CNN-BERT † & 33.59 & 37.16 & 30.95 & 30.84 & 33.20
L-GCN † & 39.01 & 37.97 & 28.81 & 26.98 & 39.10
HCRN † & 39.10 & 38.17 & 28.75 & 27.27 & 39.16
STAIR & 33.20 & 39.10 & 38.41 & 31.30 & 39.81
ClipBERT † & 39.81 & 43.59 & 32.34 & 31.42 & 43.59

Table 6: Accuracy on STAR test set, categorized by question type. †: Results from (Wu et al. 2021)

types of questions which requires spatial and temporal reasoning. Results on MSRVTT-QA shows that STAIR is also applicable to noisy, automatically-generated questions (Lin et al. 2022). However, it performs worse than the pre-trained ClipBERT and is only comparable with other simpler methods, as STAIR is designed for complex spatial-temporal reasoning while questions in MSRVTT-QA are mostly simple factoid questions.

**Conclusion**

In this paper, we propose STAIR for explainable compositional video question answering. We conduct extensive experiments to demonstrate the performance, explainability, and applicability when program annotations are not available. Moreover, STAIR is more auditable compared with previous works, it returns direct, human-understandable intermediate results for almost every reasoning step, and can be used as prompts to improve the performance of pre-trained models. We also propose intermediate supervision to improve the accuracy of intermediate results.

Possible future directions include: training program generators without direct supervision of ground truth programs (e.g., with reinforcement learning like (Mao et al. 2019)), better functional and structural designs of the neural modules (e.g., using more powerful pre-trained models), and applying on more video-text tasks other than QA.

**Acknowledgments**

This work was supported by National R&D project of China under contract No. 2022YFC3301900.

**References**


Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners.


