Video Question-Answering

Report by Aashrith Madasu (asm6590)

Project Github Repo

Section 1: Literature review and SOTA method

Video question-answering is a variant of visual question-answering where the visual input is a video. Video question-answering requires understanding of temporal dynamics in a video. This makes it more challenging than image-based VL tasks.

The current benchmarks are divided into two types:

- Factoid:
    - Focuses on objective, specific, and factual questions. Answers can typically be derived directly from visible content in a video (e.g., objects, actions, or simple events).
    - Datasets:
        - MSRVTT-QA
        - VideoQA
        - MovieQA
        - MovieFIB
        - TVQA
- Inference:
    - Demands logical reasoning, causal understanding, or even counterfactual reasoning.
    - Datasets:
        - SUTD-TrafficQA
        - PsTuts-VQA
        - V2C-QA
        - NExT-QA
        - AGQA
        - DramaQA

For the scope of this report, the following two benchmarks are selected from each category: SUTD-TrafficQA (Inference), MSRVTT-QA (Factoid)

SUTD-TrafficQA

- This dataset tests a model's understanding of video event semantics in complex traffic scenarios to be able to accurately answer questions.
- It provides 6 challenging traffic-related reasoning tasks including "Basic understanding", "Event forecasting", "Reverse reasoning", "Counterfactual inference", "Introspection", "Attribution".
- This benchmark formulates questions as multiple choice questions with only one correct answer - the number of choices is unbounded and varies from question to question.

Comparison of the state-of-the-art methods on the SUTD-TrafficQA dataset:

Method	**Accuracy**
Humans	**95.4**
VIS+LSTM	29.9
TVQA	35.2
HCRN	36.5
BERT-VQA	33.7
Eclipse	37.1
CLIP	32.3
CLIP+HCRN	41.9
ATP	35.6
Tem-adapter	**46.0**

MSR-VTT-MC

- The MSRVTT-MC (Multiple Choice) dataset is a video question-answering dataset created based on the MSR-VTT dataset.
- It consists of 2,990 questions generated from 10,000 video clips with associated ground truth captions.
- For each question, there are five candidate captions, including the ground truth caption and four randomly sampled negative choices. The objective of the dataset is to choose the correct answer from the five candidate captions.

Comparison of the state-of-the-art methods on the MSR-VTT-MC dataset:

Method	**Accuracy**
VideoCLIP	92.1
ActBERT	85.7
ClipBERT	88.2
MERLOT	90.9
CLIP	74.1
ATP	93.2
Tem-adapter	**94.3**

According to the above benchmarks, the **Tem-adapter** method performs the best. Although there are other SOTA methods which may outperform this on other benchmarks, they are not reproducible in the given time. Hence, I choose **Tem-adapter** as my SOTA method and report the reproduced results in the next section.

Link to the paper: Arxiv_Tem-adapter
Released code: Github_Tem-adapter

Section 2: Replication of the results

To replicate the results I choose the **SUTD-TrafficQA** (link) dataset. I report two types of results as follows:

Replication 1: Validation using the released model

The authors **released** their trained model as a checkpoint from the last epoch of their training. I loaded this checkpoint and ran it on the **validation set** of the SUTD-TrafficQA dataset.

Here is a comparison of the reproduced results with original results: Validation (released model)

Source	**Validation Accuracy**
Original (paper)	46
Reproduced (released model)	46

Observation: As expected, the validation results matched.

Replication 2: Training from scratch

In addition to the above, I also performed training from scratch by initializing random weights. Here is a brief overview of the comparison of training processes:

	Original training (paper)	**Replicated by me**
Epochs	50	50
Batch size	128	32
Initial LR	1e-4	1e-4
LR decay	0.5 every 10 epochs	0.5 every 10 epochs
Optimizer	Adam	Adam
GPU	single Tesla V100	single GTX 1070
Training time	8 hrs	22 hrs

Note: The LR decay is reported as a factor (<1) that is multiplied periodically.

A smaller batch size is chosen compared to what is mentioned in the paper because of the limited GPU memory capacity.

Here are the links for:

- Screenshots of the training log: Train log (scratch)
- Trained checkpoints: Ckpt_49 (scratch)

Final metrics of the training (epoch 50):

Sum loss	Avg loss	CE loss	Recon loss	Avg acc.
0.127	0.34	33.28	0.0067	98.20

Comparison of my training results with original results: Validation log (scratch)

Source	**Validation Accuracy**
Original (paper)	46
Reproduced (from scratch)	45.37

Observation: The reproduced accuracy is **very close** to the one reported. The slight drop in accuracy can be attributed to the slight change in the **optimization process**. Although the seeds are fixed for all the possible random parameters to ensure maximum reproducibility, the batch size is smaller than what is used in the paper (reason above). Each batch of the data is used to estimate gradients at each step separately. Since the batches encountered by the models are different, the corresponding gradients and parameter updates are different. Hence, the small deviation from the original accuracy is understandable.

Section 3: Improvement

To improve the performance of the current method (Tem-aligner), I extended the current architecture by incorporating a **cross-attention layer** for better video summarization.

The current architecture (paper) performs **average pooling** over frame features to compute video-level features. Although this simple operation summarizes a video effectively, a **more sophisticated** video summarizer can possibly improve the model's performance.

Hence, a cross-attention layer is incorporated to extract additional useful information from frame embeddings using a learnable query embedding. This additional "summary" features are added to the original average pooled features thereby preserving the rich frame level representations.

Here is the proposed modification:

```
old_video_embed = mean(frame_embeds)

new_video_embed = mean(frame_embeds) + cross_attn(
                                          query=query_embed,
                                          keys=frame_embeds,
                                          values=frame_embeds)
```

The extended architecture is implemented, please refer to these code files:

- TempAligner_extended.py
- VideoSummarizer.py
- train_imp.py
- validate_imp.py

For the training, only the cross-attention layer mentioned above was trained and all other layers were **frozen**. This is done to reduce the number of learnable parameters which would reduce the training time greatly.

The training was done with the following configuration:

Epochs	Batch size	Initial LR	LR Decay	Optimizer	GPU	Training time
50	32	1e-4	0.5 every 10 epochs	Adam	single RTX A4500	4 hrs

Find the links for:

- Training log: Train log (Improvement)
- Last checkpoint (epoch 50): Ckpt_49 (Improvement)

Final metrics of the training (epoch 50) of the extended architecture:

Sum loss	Avg loss	CE loss	Recon loss	Avg acc.
0.13	1.37	136.49	0.0047	97.10

Performance of the extended architecture compared to the original: Validation log (Improvement)

Source	**Validation Accuracy**
Original (paper)	46
Extended	43.88

Observation: A drop in the validation accuracy is observed. One possible reason could be that the new cross-attention layer was not jointly learned with the rest of the model (as mentioned above). Instead only the cross-attention was learned and the rest of the model is frozen. This might have prevented the new model from learning meaningful representations.

Section 4: Future Work

The extended architecture can be trained jointly from scratch to see how it performs compared to the above.