# Reasoning with Heterogeneous Graph Alignment for Video Question Answering

Jiang, Pin, and Yahong Han (AAAI 2020)

Ahjeong Seo

Seoul National University

# Contents

# 1. 논문 소개

# 1. Introduction

## Recent efforts towards VideoQA (AS – IS)

- Inter-modality correlations
  - Uncover latent correlations between video content and words' semantics

    - Beyond rnns: Positional self-attention with co-attention for video question answering (AAAI 2019, Li et al. )
    - Progressive attention memory network for movie story question answering (CVPR 2019, Kim et al. )

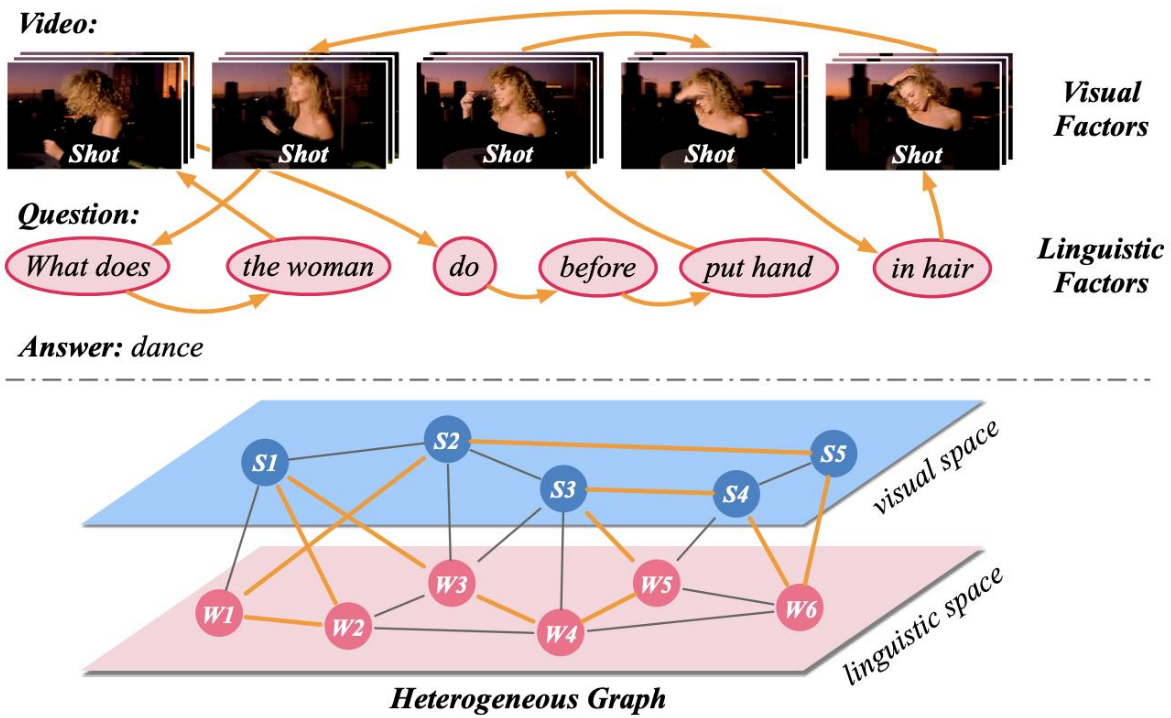
- Intra-modality correlations
  - Incorporating correlations inside videos or dependencies

    - Tgif-qa: Toward spatio-temporal reasoning in visual question answering (CVPR 2017, Jang et al. )
    - Heterogeneous memory enhanced multimodal attention model for video question answering (CVPR 2019, Fan et al. )

- Integrating correlations of both inter- and intra-modality
  - Current methods lack of simultaneously reasoning inter- and intra-modality relations

# 1. Introduction

## Heterogeneous Graph Alignment (TO – BE)




### Contributions

- Novel heterogeneous graph over different modality factors
  - Reason inside one or inter-modality

    "$s_1 \rightleftharpoons s_3$", "$s_1 \rightleftharpoons w_2$"

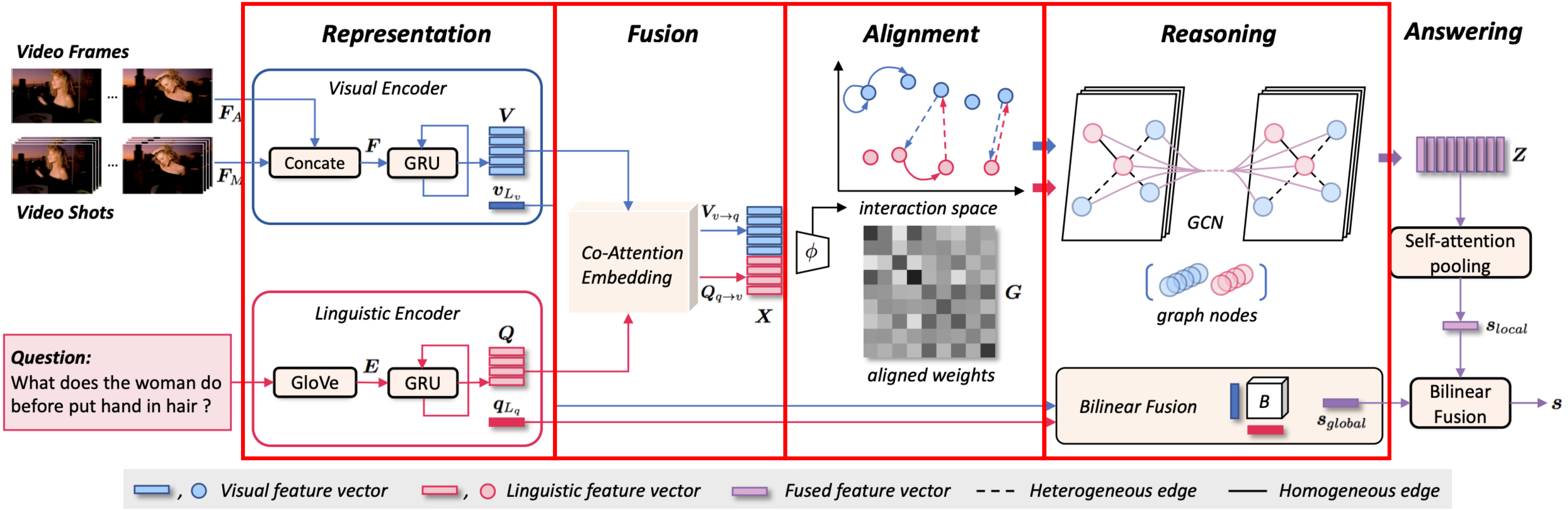
- Variety of modular co-attention embedding
  - Important role in cross-modal fusion and alignment before further aligned in GCN

# 2. Related Work

- Video Question Answering
  - Extends VisualQA to video domain
  - higher demands on spatio-temporal understanding and reasoning

- Multimodal Information Fusion
  - Early fusion(vector operation: concatenation, element-wise addition and multiplication) and projected in joint space by neural network
  - Attention mechanism enhance interaction between modalities
  - Co-attention
  - Bilinear pooling : fuse multimodal vectors by computing outer product

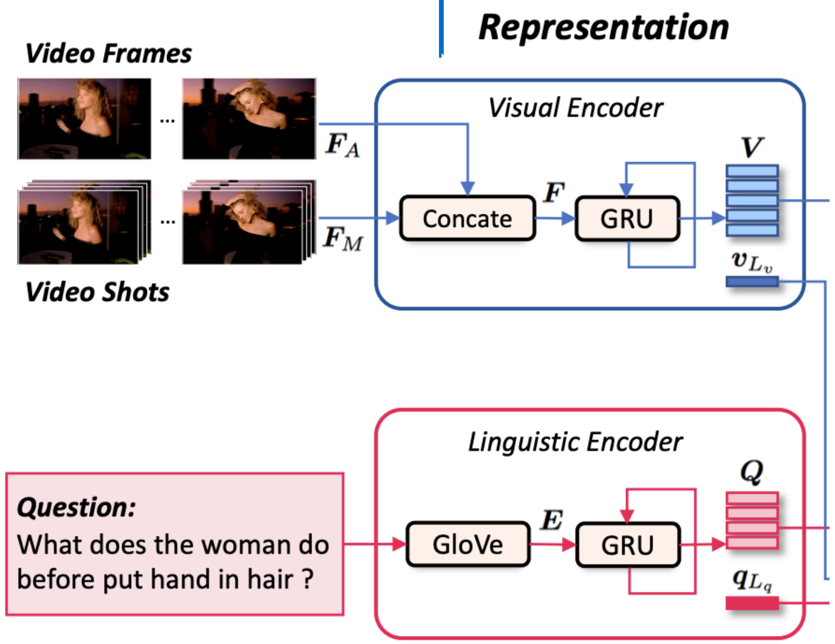
# 3. Method


Video Frames
$F_A$
$F_M$
Video Shots
Representation
Visual Encoder
Concate
$F$
GRU
$V$
$v_{L_v}$
Linguistic Encoder
GloVe
$E$
GRU
$Q$
$q_{L_q}$
Question:
What does the woman do before put hand in hair ?
Fusion
Co-Attention Embedding
$V_{v \to q}$
$Q_{q \to v}$
$X$
Alignment
$\phi$
interaction space
$G$
aligned weights
Reasoning
GCN
graph nodes
Bilinear Fusion
B
$s_{global}$
Answering
$Z$
Self-attention pooling
$s_{local}$
Bilinear Fusion
$s$
Visual feature vector
Linguistic feature vector
Fused feature vector
Heterogeneous edge
Homogeneous edge

# 3. Method

## 3.1 Visual and Linguistic Contextual Representation

**Visual Feature**




$$F_A = \{a_i : i \leq L_v, a_i \in \mathbb{R}^{d_a}\}$$ Appearance features(2D ConvNets)

$$F_M = \{m_i : i \leq L_v, m_i \in \mathbb{R}^{d_m}\}$$ Motion features(3D ConvNets)

concat / project

$$F = \{f_i : i \leq L_v, f_i \in \mathbb{R}^{d}\}$$ Joint Representation

$$V, v_{L_v} = GRU\left(F; \theta_{GRU}\right)$$ Sequence Representation

($V \in \mathbb{R}^{L_v \times d}$ : GRU output, $v_{L_v}$ : last hidden(global))

**Linguistic Feature**

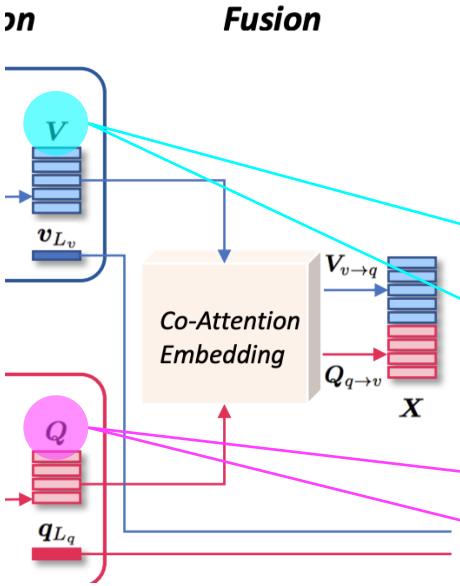
$$E = \{e_i : i \leq L_q, e_i \in \mathbb{R}^{300}\}$$ GloVe word embedding

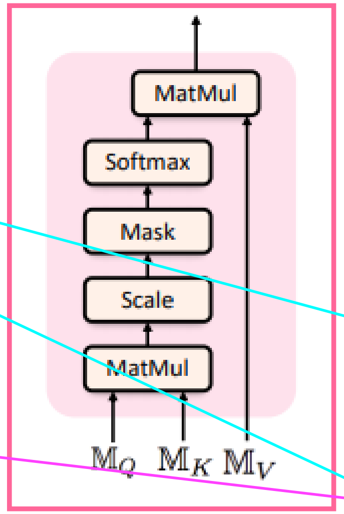
$$Q, q_{L_q} = GRU\left(E; \theta_{GRU}\right)$$ Sequence Representation

(Q $\in \mathbb{R}^{L_q \times d}$ : GRU output, $q_{L_q}$ : last hidden(global))

# 3. Method

## 3.2 Cross-Modal Joint Fusion and Alignment



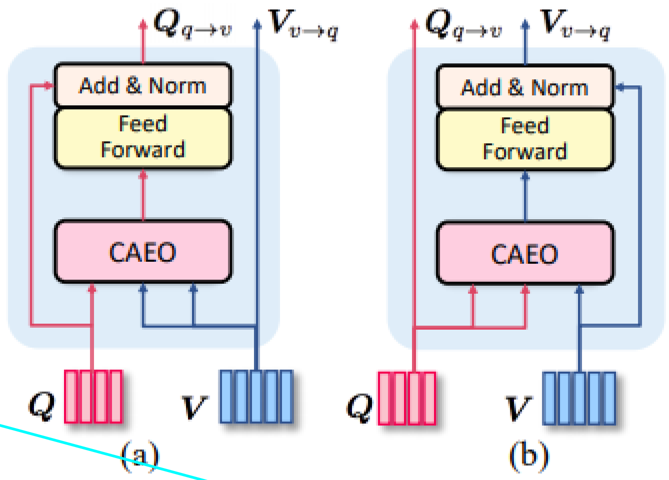




<CAEO>

$$CAEO\left(\mathbb{M}_Q, \mathbb{M}_K, \mathbb{M}_V\right) = softmax\left(\frac{\mathbb{M}_Q \mathbb{M}_K^T}{\sqrt{d}}\right)\mathbb{M}_V$$

- Query usually indicate different modality from Key, Value
- CAEO embeds the information of query into key's feature space

### 3.2.1 Attention-Based Structure




- From Linguistic to Visual space

query key/value

$$Q_{CAEO} = CAEO\left(W_q^q Q, W_k^q V, W_v^q V\right),$$

$$Q_{q\to v} = LayerNorm\left(FF_q\left(Q_{CAEO}\right) + Q\right).$$

- From Visual to Linguistic space

query key/value

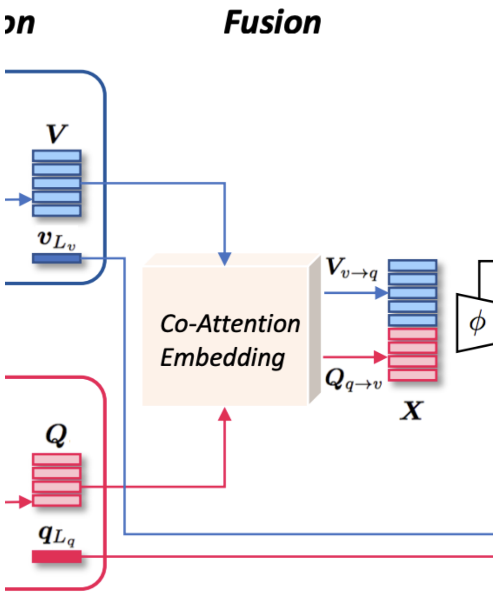
$$V_{CAEO} = CAEO\left(W_q^v V, W_k^v Q, W_v^v Q\right),$$

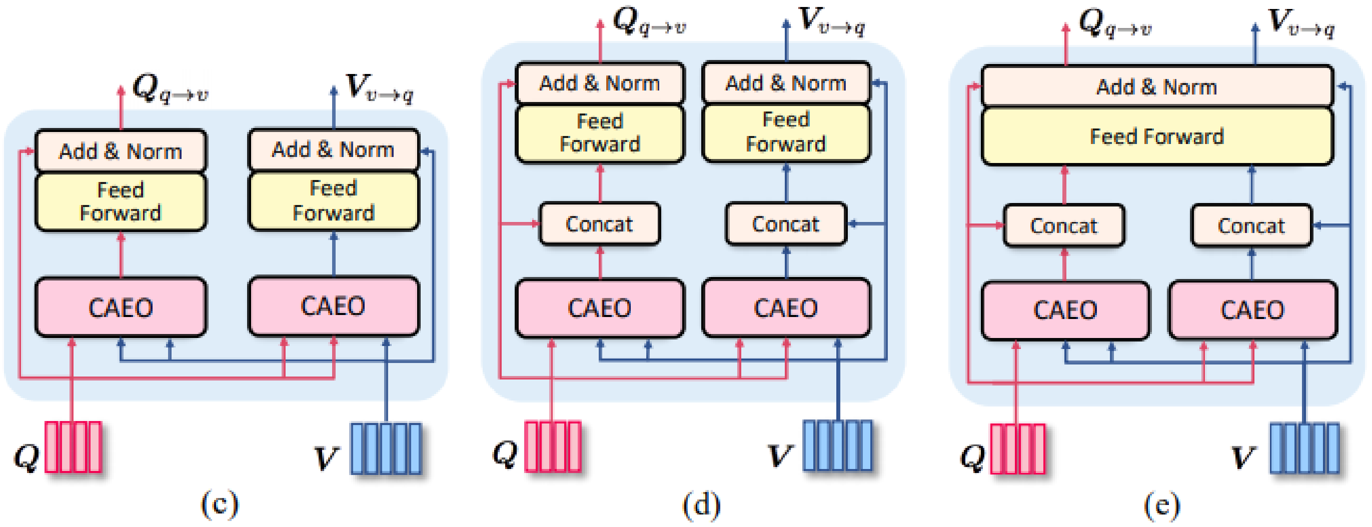
$$V_{v\to q} = LayerNorm\left(FF_v\left(V_{CAEO}\right) + V\right).$$

# 3. Method

## 3.2 Cross-Modal Joint Fusion and Alignment

### 3.2.2 Co-Attention-Based Structure




$$Q_{q\to v} = LayerNorm\left(FF_q\left([Q_{CAEO}; Q]\right) + Q\right)$$
$$V_{v\to q} = LayerNorm\left(FF_v\left([V_{CAEO}; V]\right) + V\right)$$

- Crossover transformation is crucial to fuse and align information of different modalities
- Combine two transformation and introduce a symmetrical co-attention operation

# 3. Method

## 3.3 Heterogeneous Graph Reasoning

$$X = \begin{bmatrix} Q_{update} \\ V_{update} \end{bmatrix}$$

$X \in \mathbb{R}^{N \times d}, N = L_v + L_q$

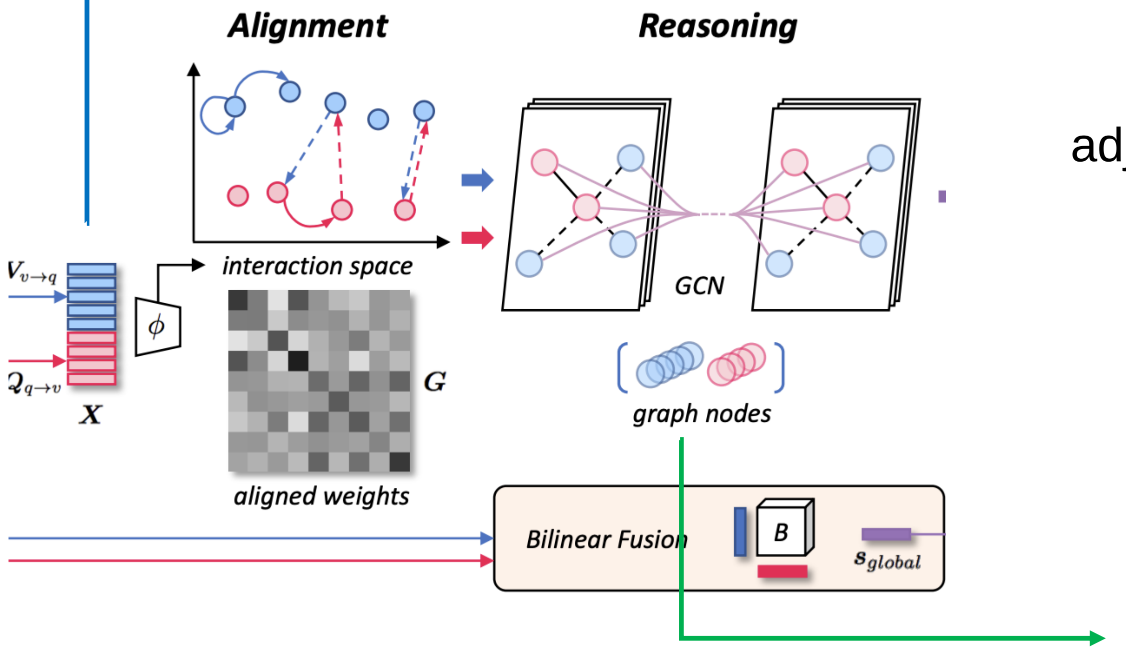
Use as graph node features

graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$

N nodes $v_i \in X$
Fully-connected edges $(v_i, v_j)$

adjacency matrix $G = \phi(X)\phi(X)^T$

1. project X into interaction space by non-linear transformation $\phi$
2. obtain correlation scores of nodes by dot-product similarity




Reasoning on GCN

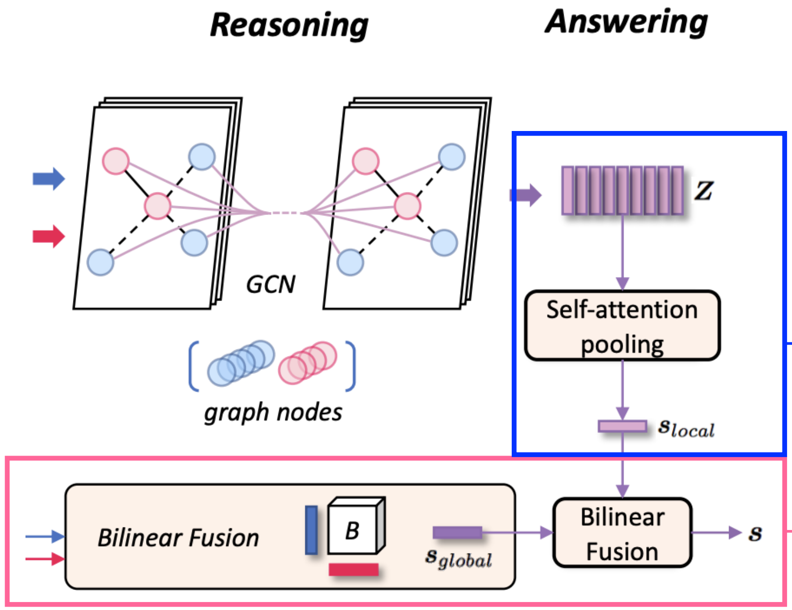
$$Z = GXW$$

Adjacency matrix $G \in \mathbb{R}^{d \times d}$
Learnable weight matrix $W \in \mathbb{R}^{d \times d}$

GCN performs relational reasoning over a graph
where each node is updated by aggregating its neighbors and itself

# 3. Method

## 3.3 Heterogeneous Graph Reasoning
## ~ 3.4 Global and Local Information Fusion




GCN output $Z = GXW$

Self-attention pooling:

ρ ≑ $FC(d) - Tanh \ - FC(1) - softmax$

$$s_{local} = \sum^{N} \rho(Z)^T Z.$$

Local vector $s_{local}$

$$s_{global} = Bilinear\left(q_{L_q}, v_{L_v}\right)$$

Global vector $s_{global}$

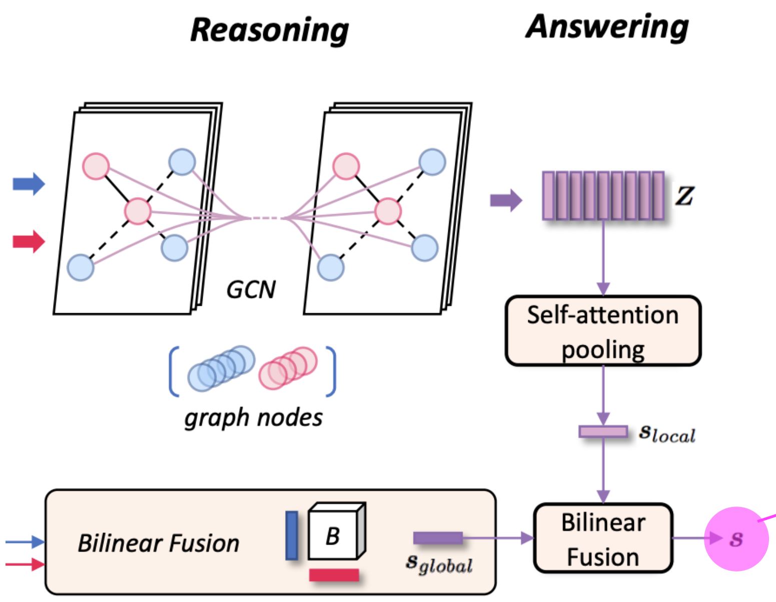
($v_{L_v}$, $q_{L_q}$ : last GRU hidden states)

Fusion

$$s = Bilinear\left(s_{global}, s_{local}\right)$$

# 3. Method

## 3.5 Answer Prediction and Evaluation




Input video clip $v \in V$, related question $q \in Q$, correct answer $a^*$ and candidate set $\{a_i\}_{i=1}^{K}$

1. Concentrate the question with each answer candidates

   Get K candidate sequences

2. Linear regression function:

   inputs the final output s

   outputs K scores for all candidates

$$Loss = \sum_{i=1}^{K-1} \max\left(0, 1 + s_i^n - s^p\right)$$

incorrect answer scores $s_i^n$

correct answer score $s^p$

# 4. Experiments

## 4.1 State of the Art Comparison

- Dataset : TGIF-QA
  - 165K Q&A pairs from 72K animated GIFs
  - Has four task types

    1) Repetition Count(*Count*) :

       open-ended numbers to count the number of repetition in action

    2) Repeating Action(*Action*) :

       multiple-choice task to identifying a repetitive action

    3) State Transition(*Trans.*) :

       5-options multiple-choice which identifies the transition of two states

    4) FrameQA(*FrameQA*) :

       open-ended task that can be answered from single frame

Table 1: State-of-the-art comparison on TGIF-QA dataset. Mean $\ell_2$ loss for Count, and accuracy (%) for others.

| Methods | Count | Action | Trans. | FrameQA |
|---|---|---|---|---|
| Random | 19.62 | 20.00 | 20.00 | 0.06 |
| ST-VQA-Sp. | 4.28 | 57.3 | 63.7 | 45.5 |
| ST-VQA-Tp. | 4.40 | 60.8 | 67.1 | 49.3 |
| ST-VQA-Sp.Tp. | 4.56 | 57.0 | 59.6 | 47.8 |
| CT-SAN | 5.14 | 56.1 | 64.0 | 39.6 |
| Co-Mem | <u>4.10</u> | 68.2 | 74.3 | 51.5 |
| PSAC | 4.27 | 70.4 | 76.9 | **55.7** |
| Fan et al. | <u>4.10</u>[1] | <u>73.9</u> | 77.8 | 53.8 |
| ST-VQA⋆ | 4.22 | 73.5 | <u>79.7</u> | 52.0 |
| Ours HGA | **4.09** | **75.4** | **81.0** | <u>55.1</u> |

# 4. Experiments

## 4.1 State of the Art Comparison

- Dataset : MSVD-QA and MSRVTT-QA
  - Two datasets generated from video descriptions
  - 50K and 243K Q&A pairs
  - Consists of different types of questions : *what, who, how, when, where*

Table 2: State-of-the-art comparison on MSVD-QA dataset. Mean $\ell_2$ loss for Count, and accuracy (%) for others.

| Methods | What (8,149) | Who (4,552) | How (370) | When (58) | Where (28) | All (13,157) |
|---|---|---|---|---|---|---|
| ST-VQA | 18.1 | 50.0 | **83.8** | 72.4 | 28.6 | 31.3 |
| Co-Mem | 19.6 | 48.7 | 81.6 | **74.1** | 31.7 | 31.7 |
| AMU | 20.6 | 47.5 | 83.5 | 72.4 | **53.6** | 32.0 |
| Fan et al. | 22.4 | 50.1 | 73.0 | 70.7 | 42.9 | 33.7 |
| Ours HGA | **23.5** | **50.4** | 83.0 | 72.4 | 46.4 | **34.7** |

Table 3: State-of-the-art comparison on MSRVTT-QA dataset. Mean $\ell_2$ loss for Count, and accuracy (%) for others.

| Methods | What (49,869) | Who (20,385) | How (1,640) | When (677) | Where (250) | All (72,821) |
|---|---|---|---|---|---|---|
| ST-VQA | 24.5 | 41.2 | 78.0 | **76.5** | 34.9 | 30.9 |
| Co-Mem | 23.9 | 42.5 | 74.1 | 69.0 | **42.9** | 32.0 |
| AMU | 26.2 | 43.0 | 80.2 | 72.5 | 30.0 | 32.5 |
| Fan et al. | 26.5 | 43.6 | 82.4 | 76.0 | 28.6 | 33.0 |
| Ours HGA | **29.2** | **45.7** | **83.5** | 75.2 | 34.0 | **35.5** |

# 4. Experiments

## 4.2 Ablation Study

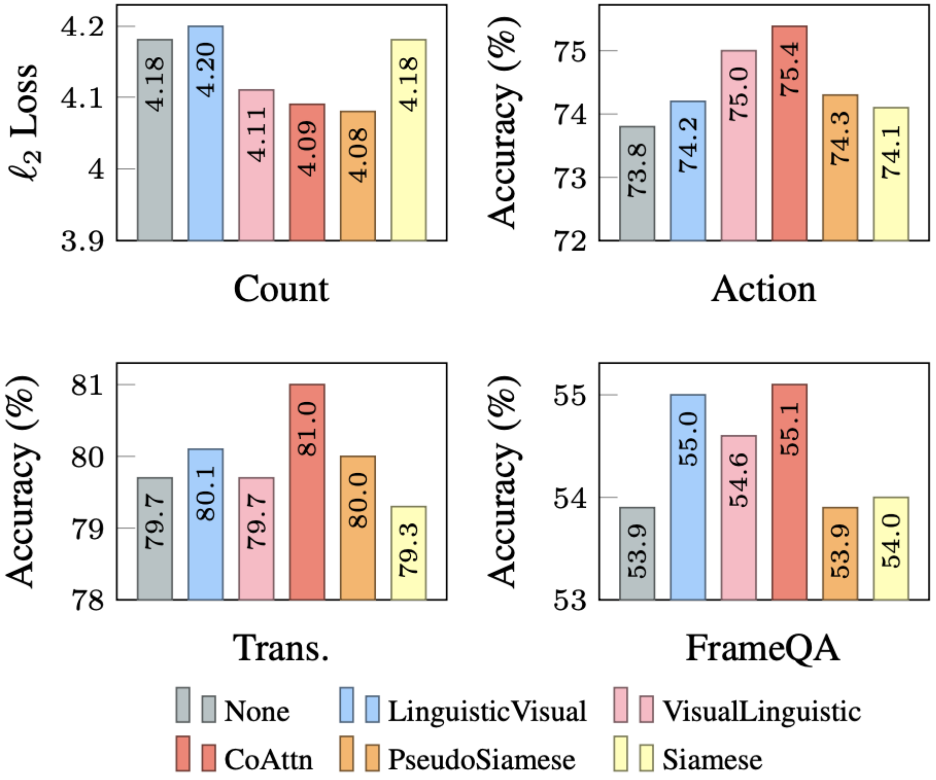



Figure 4: Experimental results of *CAEO* variants.

- On TGIF-QA dataset
- Appending CAEO improve the performance to some extent
- CoAttn achieves the best on *Action, Trans.*, and *FrameQA*
- But sub-optimal in *Count* task
- PseudoSiamese and Siamese hurt model quality

# 4. Experiments

## 4.2 Ablation Study

Table 4: Ablation study on TGIF-QA dataset. Mean $\ell_2$ loss for *Count*, and accuracy (%) for others.

| Methods | Count | Action | Trans. | FrameQA |
|---|---|---|---|---|
| GRU (w/ local fusion) | 4.31 | 55.4 | 69.8 | 53.0 |
| + global fusion (baseline) | 4.19 | 73.4 | 78.1 | 55.7 |
| + GCN §3.3 | 4.18 | 74.5 | 79.7 | 53.9 |
| + CoAttn §3.2 | 4.17 | 73.9 | 78.8 | 55.4 |
| + GCN §3.3 | 4.09 | 75.4 | 81.0 | 55.1 |
| HGA w/o global fusion | 4.25 | 73.6 | 79.6 | 53.6 |
| HGA w/o local fusion | 4.24 | 71.3 | 77.8 | 53.9 |

- On TGIF-QA dataset
- Methods
  - GRU : vanilla parallel architecture
  - + global fusion : in section 3.4 (add local feature with global feature)
  - + GCN : add GCN
  - + CoAttn : add CAEO module
  - + GCN : add GCN and CAEO
- Combining "CoAttn" and "GCN"

  → Two modules promote each other

# 4. Experiments

## 4.3 Examples of successful cases

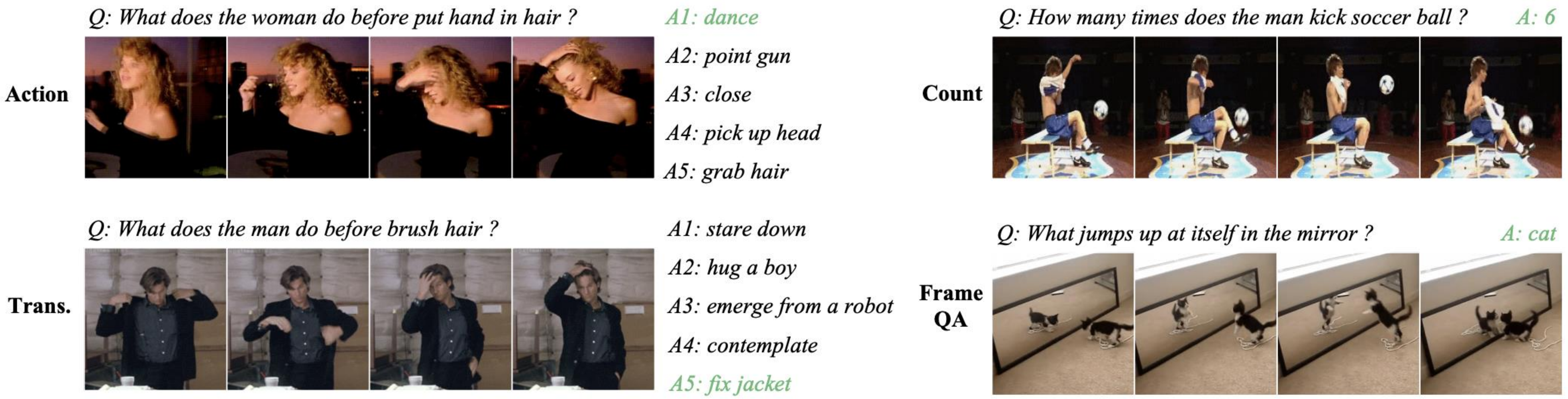



Figure 5: Typical examples of successful cases on TGIF-QA dataset.

# 2. Reproduce 결과

## ▪ TGIF-QA Dataset에 대해 성능 비교 결과

### 1. 논문에 report된 성능

| Methods | Count | Action | Trans. | FrameQA |
|---|---|---|---|---|
| HGA | 4.09 | 75.4 | 81.0 | 55.1 |

### 2. Reproduce 성능

| Methods | Count | Action | Trans. | FrameQA |
|---|---|---|---|---|
| Reproduced HGA | 4.05 | 75.50 | 80.95 | 55.24 |

- Count와 Action, FrameQA의 경우 report된 성능보다 오히려 높은 accuracy를 얻었습니다.
- 주어지는 test dataset에 대해 실험하였기 때문에 실제 leaderboard상에서의 성능과 차이를 보이는 것으로 예상됩니다.

- **Training log는 github code의 log 폴더에 저장되어 있습니다.**

- Inference 결과 화면 캡쳐

- Count

```
Test|Epoch: 0, Acc: 23.747890=844/3554, Test Loss: 3.965284
Test|Count Real Loss:    4.052
```

- Action

```
Test|Epoch: 0, Acc: 75.505717=1717/2274, Test Loss: 0.424502
Save model at  ./saved_models/MMModel/
```

- Trans.

```
Test|Epoch: 0, Acc: 80.953145=5045/6232, Test Loss: 0.211618
Save model at  ./saved_models/MMModel/
```

- FrameQA

```
Frame Frame
Test|Epoch: 0, Acc: 55.240669=7563/13691, Test Loss: 2.224057
Save model at  ./saved_models/MMModel/
```

# 3. 코드 실행 방법

- Github Link: https://github.com/ahjeongseo/HGA

1. Requirements

    - python 3.6
    - pytorch 1.1
    - colorlog
    - bootstrap.pytorch
    - block.bootstrap.pytorch
    - numpy==1.18.5

(설명)

1. ‘conda create –n final_project python=3.6’ 명령어로 python 3.6 가상환경을 만듭니다.
2. ’conda install pytorch==1.1.0 torchvision==0.3.0 cudatoolkit=9.0 -c pytorch’ 로 pytorch 1.1을 설치합니다.
   저의 경우 cuda 9.0 version을 사용하였습니다. Cuda version에 따라 https://pytorch.org/get-started/previous-versions/를 참고할 수 있습니다.
3. ‘pip install -r requirements.txt’를 통해 필요한 패키지를 다운 받습니다.

## 2. Data Download

- C3D feature, ImageNet feature, GloVe feature:

  https://github.com/fanchenyou/HME-VideoQA/tree/master/gif-qa에서 다운로드

- TGIF QA question csv file:

  https://github.com/YunseokJANG/tgif-qa/tree/master/dataset에서 다운로드

## 3. Data path 설정

- main.py에서 주석처리한 부분 path설정

  1) feat_dir : ImageNet 및 C3D feature를 저장한 path

  2) vc_dir : GloVe feature를 저장한 path, word embedding관련 파일이 dataloader를 실행하면 저장됨

  3) df_dir : TGIF question csv file을 저장한 path

  4) checkpoint : model inference시 model file명

  5) save_path : training시 model 저장 path 및 inference시 모델 불러올 path

# 4. Training

**CUDA_VISIBLE_DEVICES=0 python main.py --test --task Count --num_workers 2 --batch_size 64**

- Task : Count, Action, Trans, FrameQA

# 5. Test (Inference)

**CUDA_VISIBLE_DEVICES=0 python main.py --test --task Count --num_workers 2 --batch_size 64**

- Task : Count, Action, Trans, FrameQA
- checkpoint option추가 가능 : inference 할 모델 파일 명 추가 (ex. –checkpoint Count_4.092.pth)

# 6. 중요 코드 설명

- /data_utils/dataset.py : TGIFQA dataloader를 구성하기 위한 class 파일
- /data_utils/.. : 나머지 파일은 dataset.py를 구성하기 위한 submodule

- /models/lstm_cross_cycle_gcn_dropout.py : HGA 모델이 구현된 class 파일
- lstm_cross_cycle_gcn_dropout.py의 submodule들
  - /models/rnn_encoder.py : video 및 text의 sequence representation을 위한 rnn 구현
  - /models/q_v_transformer.py : video 및 text의 attention, co-attention 구현
  - /models/gcn.py : co-attention된 video, text feature를 gcn으로 학습하기 위한 코드 구현

- main.py : 모델 학습을 위한 training 코드

* *모델 부분 코드 구현(models/…py)에 대한 자세한 설명은 주석으로 추가하였습니다.*