COL775: Deep Learning

Project: Temporal and Causal Reasoning from Videos on CLEVRER

Aman Bhardwaj (2022ANZ8400) **Aniruddha Deb (2020CS10869)**

Team: Bay(es)watch

Neuro-symbolic reasoning on videos uses neural networks and symbolic reasoning to understand from videos. It uses deep learning models for video comprehension and symbolic reasoning to connect low-level visual perception and high-level semantic understanding. This multidisciplinary approach lets the system distinguish objects, activities, and events in movies and interpret their contexts and implications. Combining neural networks and symbolic reasoning can improve video analysis, comprehension, summarization, captioning, and question answering.

1 Pre-processing Details

1.1 Video pre-processing:

1. **Frames Extraction:**t In order to make extract features from the video, we extract all the frames from the video. CLEVRER videos have been designed at 24FPS. Therefore, each video has 128 frames in total.
2. **Frames Normalization:** We first calculate the new *Mean & Std* for the video frames by random sampling of 128K frames.

   $\mu = (129.2153,\ 128.6306,\ 127.7490),\ \sigma = (14.3266,\ 14.8389,\ 15.6828)$

1.2 Descriptive Question Pre-processing:

For descriptive question we append the `question_subtype` at the end of the question in order to provide additional context to the text encoder about the type of answer it has to predict.

$$[DQ_i] = [CLS] + [question] + [SEP] + [question\_subtype]$$

1.3 Non-Descriptive Question Pre-processing:

For non-descriptive question we experiment with two different schemes.

- **Scheme-1:** For instance `Q1` has 2 choices, therefore, (`question, choice_1, choice_2`) has been divided into two separate question.

$$[Q1] = [CLS] + [question] + [SEP] + [choice\_1],\ [Q2] = [CLS] + [question] + [SEP] + [choice\_2]$$

- **Scheme-2:** For instance `Q1` has 2 choices, therefore, (`question, choice_1, choice_2`) has been divided into two separate question.

$$[Q] = [CLS] + [question] + [SEP] + [choice\_1] + [SEP] + [choice\_2]$$

2 Baseline-1: CNN + BERT

2.1 Model Architecture and Experimentation

Figure 1 illustrates the model architecture block diagram for CNN+BERT baseline model. Our architecture has three main blocks: 1. Video Encoder 2. Text Encoder 3. Classification Task Heads. Corresponding details have been discussed as follows.

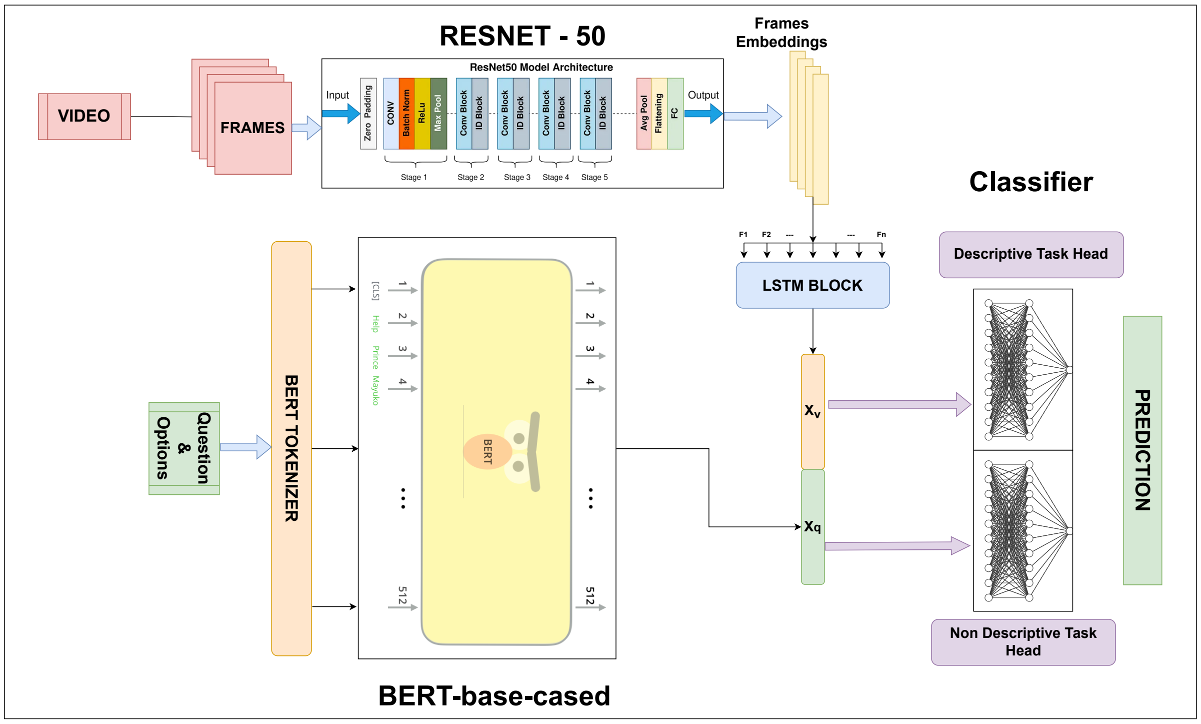

Figure 1: Model Architecture of CNN+BERT Baseline Model.

2.1.1 Video Encoder

1. **ResNet Block:** To extract the features from the frames.
2. **LSTM Block:** To embed temporal information about the encoded frame, we employ uni-directional LSTM block. We used **last hidden state h_T** as the video embedding.

2.1.2 Text Encoder:

As specified in the problem statement we employ BERT-base-cased as the text encoder.

2.1.3 Classification Task Heads:

Here we experiment with two different settings.

1. **Setting-1:** Only two taskheads: Descriptive and Non-Descriptive.
2. **Setting-2:** Four separate taskheads: Descriptive, Explanatory, and Counterfactual, Predictive task heads.

2.1.4 Libraries Used:

1. `BertModel`: Huggingface
2. `bert-base-cased tokenizer`: Huggingface
3. `ResNet`: Torchvision
4. `torch`

2.1.5 Experimentation:

We experiment with the following configurations:

1. **Num_Frames sampled per video:** Uniformly sampling of 25 frames from the video enabled us to forward-pass one whole video at a time through ResNet, while all 128 all a time were resulting in `OOM` error.
2. **ResNet-34 and ResNet-50:** We couldn't try resnet-101 due to gpu-memory limitation.
3. **Classification heads:** As described in pre-processing we experiment with two different settings of classification heads.
4. **Question and Options Encoding:** As illustrated in pre-processing.
5. **Full/partial Tuning:** Initially we kept all layers of resnet and BERT tunable end-to-end. However, the training speed was extremely slow in this case `(1.5its/sec)`. To speed up the process we freeze the weights of first few layers of ResNet and BERT which enabled us with training speed of `(3.8its/sec)`.
6. **Different learning-rates:** We experiment with different learning rates.

2.2 Results and Inferences

Table 1 contains the performance metrics of baseline-1 CNN+BERT.

Table 1: Performance of Baseline-1: ResNet+BERT model on CLEVRER dataset.

Model	Descriptive	Explanatory		Predictive		Counterfactual		Avg. Per Ques	Description
		Per-Choice	Per-Ques	Per-Choice	Per-Ques	Per-Choice	Per-Ques		
CNN+BERT	**36.73%**	**62.71%**	**13.27%**	**50.90%**	**16.42%**	**53.28%**	0.37%	16.70%	4 TaskHeads
CNN+BERT	27.74%	53.22%	12.57%	50.78%	12.57%	52.27%	**12.57%**	NA	2 TaskHeads (valset)

2.2.1 Inferences

1. **ResNet Models:** We experiment with ResNet-34, 50, 101. ResNet 50 gives better performance on than 34. But we couldn't proceed further with ResNet-101 because it was giving `OOM` memory.
2. **TaskHeads Comparison:** Model with 4 Taskheads for all four question types performs better than the 2 Taskheads namely descriptive and non-descriptive.

3. The model with 4 Taskheads shows improvements on all the question types but Counterfactual.

4. **Overall Performance:** ResNet+CNN shows predictive performance on CLEVRER dataset. However, the performance is sub-optimal as compared to the leaderboard. This can be because we are not using any information from the annotations which could be utilized in a physics engine.

3 Baseline-2: VideoCLIP + BERT

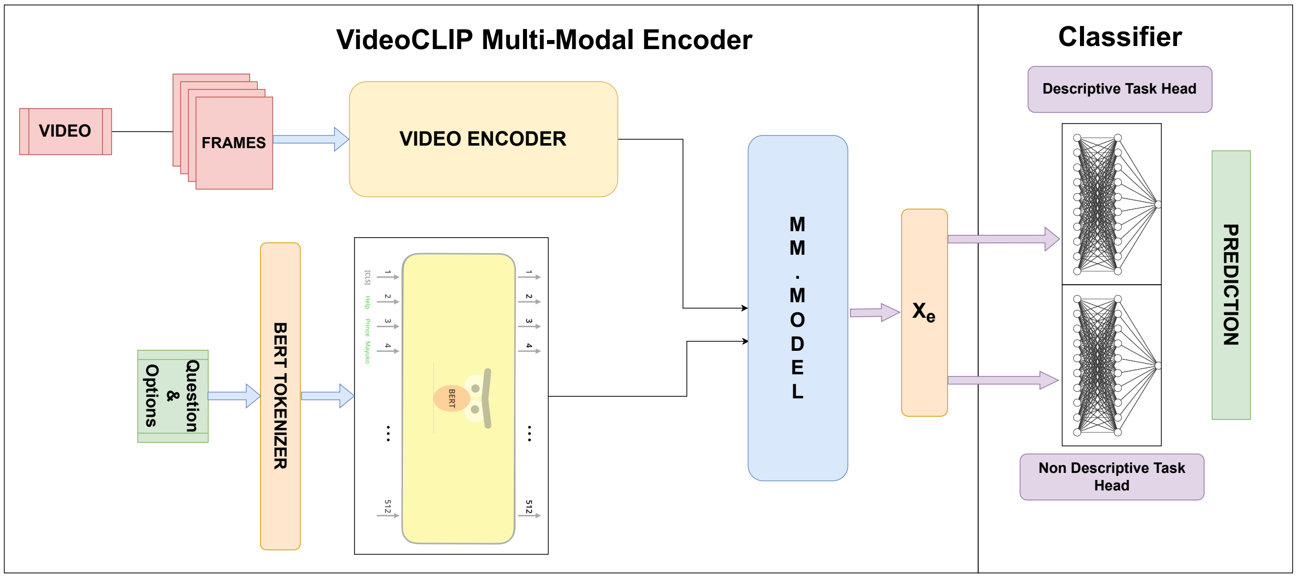

Figure 2: Model Architecture of VideoCLIP Baseline Model.

3.1 Model Architecture and Experimentation

Figure 2 illustrates the block diagram of architecture employed for the implementation of baseline-2. It comprises of two major blocks: 1. Multi-modal video and text encoder as **VideoCLIP** and 2. Classification Block with different task heads.

3.1.1 Multi-modal Video and Text Encoder:

As illustrated in Figure 2, **VideoCLIP** constitutes of three major modules:

1. **Video Encoder:** It takes video input as `(B, T, FPS, H, W, C)` where, B=BatchSize, T=Seconds, FPS=FrameRate, and H,W,C = Frame dimensions.

2. **Text Encoder:** It uses `BERT-base Model` as text encoder.

3. **Multi-Modal Model:** Video + Question embeddings are then fed to Multi-model Model to provide joint embeddings for Video and associated text.

3.1.2 Classification Task Heads:

Here we experiment with two different settings.

1. **Setting-1:** Only two taskheads: Descriptive and Non-Descriptive.

2. **Setting-2:** Four separate taskheads: Descriptive, Explanatory, and Counterfactual, Predictive task heads.

3.1.3 Libraries Used:

1. `VideoCLIP`: Fairseq MMPT library, Apex (for faster training)
2. `Pytorch`

3.2 Experimentation

1. **Classification heads:** As described in pre-processing we experiment with two different settings of classification heads.
2. **Question and Options Encoding:** As illustrated in pre-processing.
3. **Different learning-rates:** We experiment with different learning rates.

3.3 Results and Inferences

Table 2 contains the results of experimentation done for baseline-2 VideoCLIP.

Table 2: Performance of Baseline-2: VideoCLIP model on CLEVRER dataset.

Model	Descriptive	Explanatory		Predictive		Counterfactual		Avg. Per Ques	Description
		Per-Choice	Per-Ques	Per-Choice	Per-Ques	Per-Choice	Per-Ques		
VideoCLIP	45.91%	63.85%	15.78%	50.95%	21.73%	53.68%	5.37%	22.20%	2 TaskHeads
VideoCLIP	46.86%	64.21%	19.98%	52.15%	25.01%	54.97%	9.20%	25.26%	4 TaskHeads 7 Epochs
VideoCLIP	**48.35%**	**65.64%**	**25.02%**	**55.43%**	**30.15%**	**56.14%**	**12.30%**	**28.96%**	4 TaskHeads 10 Epochs

3.3.1 Inferences

1. **Taskheads Comparison:** Similar to baseline 1, this also shows that the performance of four different task heads is better than the two.
2. **Comparison with baseline-1:** Baseline 2 shows significant improvement over baseline-1. This is possibly because of multi-model model employed by MMPT to train on videos and questions together, resulting in learning better representations.
3. **Comparison with leaderboard:** Here as well, the performance is decent on CLEVRER dataset, but far behind the models on the leaderboard.

4 Competitive Part:

For the competitive part, we explored a few Visual Reasoning libraries apart from the two baselines namely: 1. VRDP 2. SlotFormer.

However, due to broken links of VRDP library, we decided to move further with SlotFormer. We trained the model from scratch. The details have been provided in the following sub-section:

4.1 SlotFormer Model Architecture

Figure 3 [1] illustrates the model architecture for SlotFormer model architecture.

It constitutes of different modules namely 1. Encoder, 2. Decoder, 3. Object-centric representation module, 4. Contrastive learning module, 5. Generator, and 6. Discriminator, which work in a sequential manner. The steps performed by SlotFormer have been listed below. The SlotFormer architecture can be summarized as follows. Given a sequence of multiple video frames as input.

[1]The figure has been taken from SlotFormer's webpage

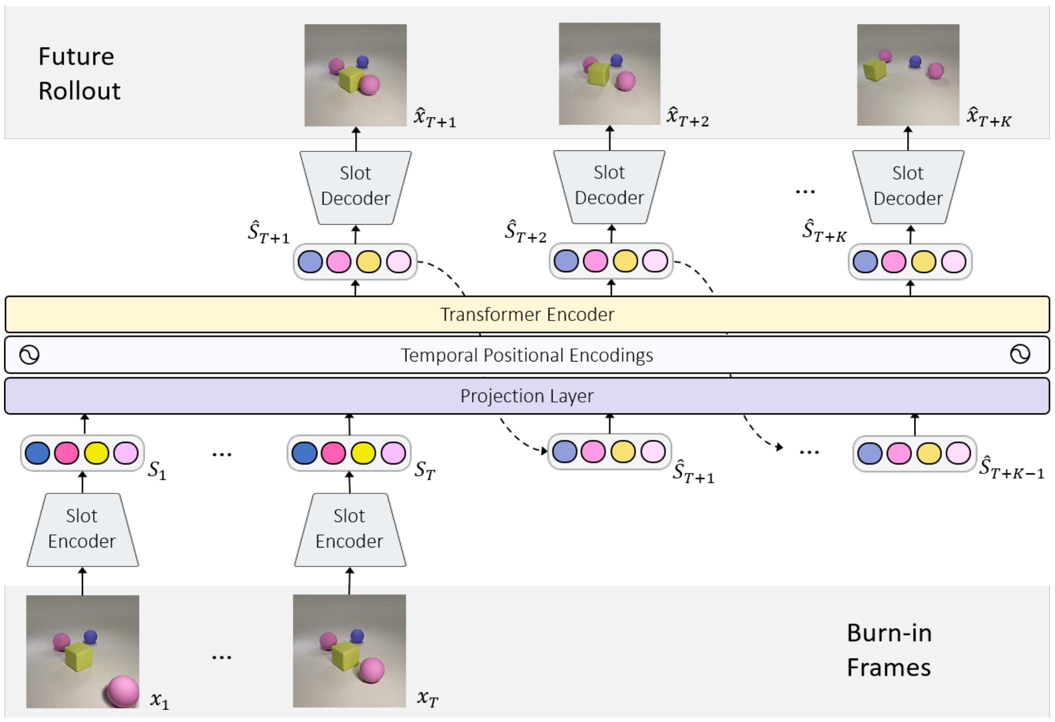

Figure 3: Model Architecture of SlotFormer.

1. The first step is to extract object slots using a pre-trained object-centric model. These slots represent individual objects in the scene.
2. The slots are transformed by applying linear projection and temporal positional encoding. This prepares the tokens for processing by the Transformer module.
3. The Transformer module then generates future slots in an auto-regressive manner. Each future slot is generated based on the previously generated slots.

4.2 Pre-processing Details

As a part of pre-processing, we perform the restructuring of data directory to make it inline with the input of SlotFormer.

4.3 Model Training Details

It includes following steps:

1. Training of `SAVi`.
2. Generation of Slots using SAVi model weights.
3. Train a SlotFormer model on extracted slots.
4. Rollout the slots
5. For video question answering, training of Aloe model on rolled out slots

The detailed guidelines for model training from scratch has been provided on the Github page of SlotFormer (link).

4.4 Results

Table 3: Performance of competitive part on CLEVRER dataset.

Model	**Descriptive**	**Explanatory**		**Predictive**		**Counterfactual**		**Avg. Per Ques**	**Description**
		Per-Choice	**Per-Ques**	**Per-Choice**	**Per-Ques**	**Per-Choice**	**Per-Ques**		
SlotFormer	94.25%	97.16%	92.00%	91.47%	84.17%	89.48%	70.83%	85.31%	NA

Table 3 contains the results of SlotFormer on CLEVRER dataset for the competitive part of the assignment.

****