VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

DATA REPRESENTATION

COURSE NAME: STATISTICAL MACHINE LEARNING

Student:
Vu Minh Phat (21127739)
Trieu Gia Huy (22127166)
Vo Thanh Nghia (22127295)
Vo Thinh Vuong (22127464)

Lecturer:
Dr. Ngo Minh Nhut
TA: Le Long Quoc

August 9, 2025

Contents

1 Information

1.1 Student Information

Student ID	Full name	Email
22127166	Trieu Gia Huy	tghuy22@clc.fitus.edu.vn
22127295	Vo Thanh Nghia	vtnghia22@clc.fitus.edu.vn

1.2 Source Code

2 Abstract

3 Model Architecture

This section describes the proposed Bidirectional Transformer with Knowledge Graph (BTKG) model [1] (Figure 1). We begin by reviewing the basic transformer module and how video features are extracted and captions generated. We then detail the BTKG architecture, including the Spatio-Temporal Encoder, the Objects and Relationships Encoder, and the bidirectional decoder (Backward and Forward Decoders). Finally, we explain the training strategy, including pseudo reverse captions and the loss functions.

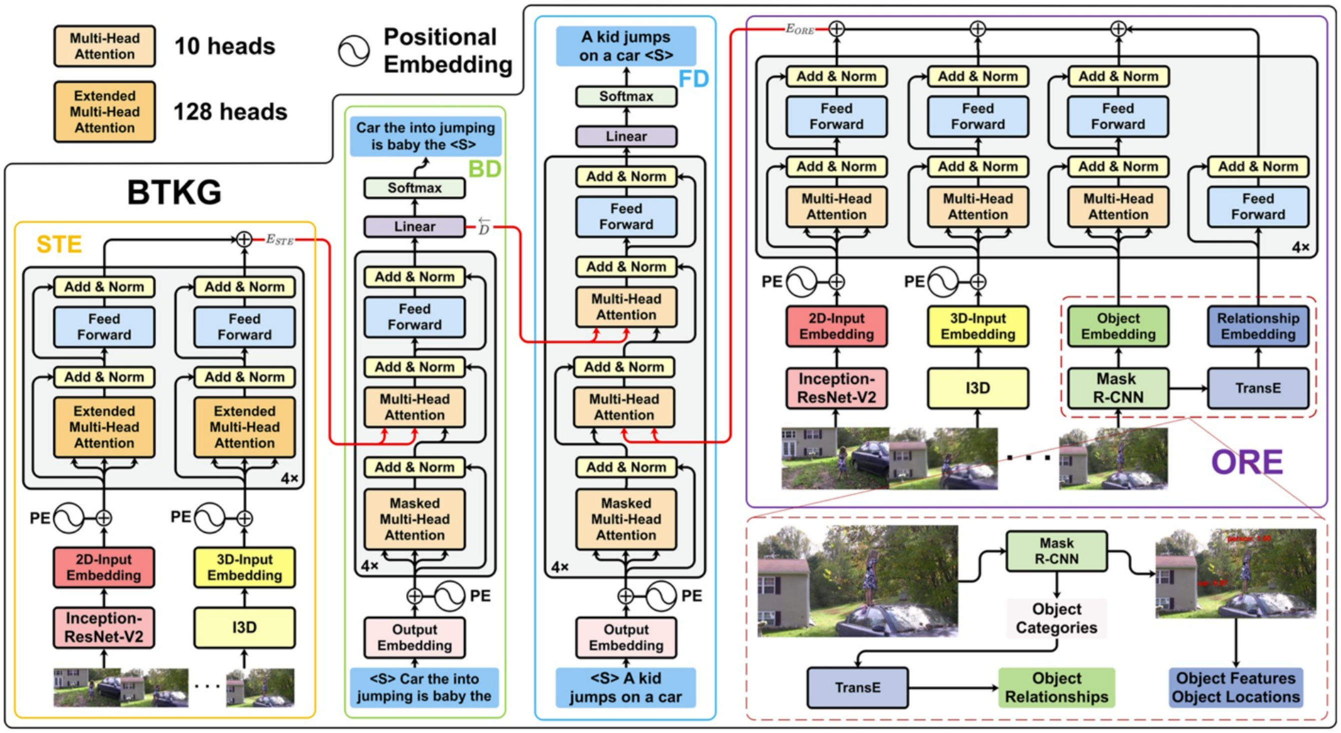


Figure 1: The figure illustrates the proposed BTKG based on the transformer architecture in detail, which consists of Spatio-Temporal Encoder (STE), Objects and Relationships Encoder (ORE), Backward Decoder (BD), and Forward Decoder (FD).

3.1 Basic Module

3.1.1 Transformer Architecture

The model is built upon the standard Transformer [2] architecture. A Transformer consists of an encoder and a decoder, each formed by stacking identical layers. Within each layer are two sub-layers: a multi-head attention (MHA) mechanism and a position-wise feed-forward network (FFN), each followed by residual connections and layer normalization. In the MHA sub-layer, the input queries (Q), keys (K), and values (V) are linearly projected into multiple "heads". Specifically, for head i, we compute QW_i^Q, KW_i^K, and VW_i^V, where W_i^Q, W_i^K, W_i^V are learned projection matrices.

The attention outputs of the h heads are concatenated and projected via W^O:

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O, \\ \text{head}_i &= \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V).\end{aligned} \tag{1}$$

The Attention function uses scaled dot-product attention:

$$\text{Attention}(Q_i, K_i, V_i) \ = \ \text{softmax}\Big(\frac{Q_i K_i^T}{\sqrt{d_k}}\Big)\, V_i,$$

where d_k is the dimensionality of the key vectors.

The FFN sub-layer applies two linear transformations with a ReLU nonlinearity in between. For an input X,

$$\text{FFN}(X) = \max(0,\ XW_1 + b_1)\, W_2 + b_2,$$

with parameters W_1, W_2, b_1, b_2. Together, these MHA and FFN components provide the basic sequence modeling capability of the Transformer.

3.1.2 Multimodal Features Extraction and Captions Generation

The task is video captioning, so the model must encode both visual and dynamic information. To this end, the system extracts **image features** and **motion features** from the video. It uses the Inception-ResNet-V2 network (pretrained on ImageNet) to extract per-frame image features, capturing static appearance. For motion, it uses the Inflated 3D ConvNet (I3D) pretrained on the Kinetics dataset to extract features from sequences of frames (capturing temporal dynamics). These networks produce feature sequences $F_I = \{f_1, \ldots, f_L\}$ and $F_M = \{v_1, \ldots, v_N\}$, where $f_i \in \mathbb{R}^{d_I}$ are image features for L sampled frames and $v_j \in \mathbb{R}^{d_M}$ are motion features (with $N = L/64$ in practice). Both feature sequences are then projected into a common model dimension d_{model} via learned linear mappings.

In addition to these frame-level features, the model incorporates **object-level semantics**. A Mask R-CNN detector (trained on the COCO dataset) detects objects in each frame and produces object feature vectors and class labels. For each detected object, its class name (e.g. "car", "person") is used in a knowledge graph module: a TransE-based model predicts relationships (triplets) between object categories. The object labels and their predicted relations are embedded via a shared word-embedding matrix (e.g. Word2Vec) to produce relation features.

For caption generation, this base module uses a bidirectional decoding strategy [3]: there is a backward decoder and a forward decoder. The backward decoder generates captions right-to-left (reverse order), while the forward decoder generates left-to-right. This bidirectional decoder takes as input the encoded video features. The backward decoder's output (a reverse caption and its hidden context) is fed into the forward decoder to improve forward captioning. In summary, the basic module provides: a Transformer backbone, multimodal feature encodings (image, motion, objects, relations), and a bidirectional decoding scheme.

3.2 Bidirectional Transformer with Knowledge Graph (BTKG)

The BTKG model integrates the above components into a unified architecture. It connects a bidirectional decoder with an encoder that incorporates knowledge-graph-enriched object features. As shown in Figure 1, BTKG consists of four parts:

- **Spatio-Temporal Encoder (STE):** encodes fine-grained frame and motion features to produce E_{STE}.
- **Objects and Relationships Encoder (ORE):** encodes object features and their relationships (along with coarse video features) to produce E_{ORE}.
- **Backward Decoder (BD):** generates a reverse caption Y_{BD} (right-to-left) and outputs a hidden context $\overleftarrow{D}$.
- **Forward Decoder (FD):** generates the final caption Y_{FD} (left-to-right), integrating both E_{ORE} and the reverse context $\overleftarrow{D}$.

The next subsections explain each component in detail.

3.2.1 Spatio-Temporal Encoder (STE)

The STE is designed to capture the fine-grained spatio-temporal content of the video. It operates on the image and motion feature sequences extracted from the video frames (as described above). Formally, let $X = \{x_1, \ldots, x_L\}$ be the sampled frames. We extract image features

$$F_I = \{f_1, \ldots, f_L\}, \quad f_i \in \mathbb{R}^{d_I},$$

and motion features

$$F_M = \{v_1, \ldots, v_N\}, \quad v_j \in \mathbb{R}^{d_M}, \quad N = L/64,$$

using Inception-ResNet-V2 and I3D respectively. To combine these modalities, each feature vector is linearly projected to a common dimension d_{model} by learned weights:

$$f_i' = W_L f_i + b_L, \quad v_j' = W_N v_j + b_N,$$

where $W_L \in \mathbb{R}^{d_{\text{model}} \times d_I}$, $W_N \in \mathbb{R}^{d_{\text{model}} \times d_M}$, and biases $b_L, b_N \in \mathbb{R}^{d_{\text{model}}}$. This yields transformed sequences $F_I' = \{f_1', \ldots, f_L'\}$ and $F_M' = \{v_1', \ldots, v_N'\}$, where $f_i', v_j' \in \mathbb{R}^{d_{\text{model}}}$.

The sequences F_I' and F_M' are then each fed through a Transformer encoder (with self-attention and FFN sublayers) to produce embeddings $E_I \in \mathbb{R}^{L \times d_{\text{model}}}$ and $E_M \in \mathbb{R}^{N \times d_{\text{model}}}$. Finally, these are fused by element-wise addition to produce the STE output:

$$E_{STE} = E_I \oplus E_M,$$

where $\oplus$ denotes element-wise sum. Thus $E_{STE} \in \mathbb{R}^{L \times d_{\text{model}}}$ encodes frame-level (fine-grained) video content by blending appearance and motion features.

A notable detail is that in the STE, the model uses an **extended multi-head attention** with 128 heads (instead of the usual 8). Using more heads allows the model to capture correlations among positions at a finer temporal scale than whole frames. In practice, the STE employs $h = 128$ attention heads as indicated in Eq. 1, which helps BTKG capture fine temporal dependencies.

3.2.2 Objects and Relationships Encoder (ORE)

The ORE captures higher-level semantics by encoding detected objects and their relationships. First, objects are detected in the video using Mask R-CNN (with a Feature Pyramid Network and ROI pooling). For each detected object i (up to k objects), we record its normalized bounding box coordinates and a visual feature vector. Let $R_l = [l_1, \dots, l_k]$ be the list of object locations (each l_i is 4-dimensional) and $R_v = [v_1, \dots, v_k]$ the list of object feature vectors. These are concatenated per-object: the i-th object feature is $(l_i|v_i)$, and stacking these forms the object feature matrix F_o:

$$F_o = \text{concat}_{i=1}^{k}\big(l_i,\, v_i\big).$$

in the paper's notation this is written as

$$F_o = \text{concat}_{i=1,\dots,k}(R_l\, R_v),$$

meaning the row-wise concatenation of R_l and R_v for each object.

In parallel, the object class labels (one-hot vectors o_i) are input to the TransE knowledge graph model to predict relations between objects. This yields a set of relation one-hot vectors $\{r_1, \dots, r_m\}$, where $m = \binom{k}{2}$. Both the object labels o_i and relation labels r_j are embedded into the same vector space via a shared embedding matrix W (pretrained by Word2Vec). Specifically:

$$o'_i = o_i W, \quad r'_j = r_j W,$$

producing embedded matrices $R'_o = [o'_1, \dots, o'_k]$ and $R'_r = [r'_1, \dots, r'_m]$. These are concatenated vertically to form the relation feature matrix F_r:

$$F_r = \begin{bmatrix} R'_o \\ \\ R'_r \end{bmatrix}$$

The matrix F_r thus contains semantic features of objects and predicted relations.

Next, the ORE encodes the content of F'_I, F'_M (the same projected image/motion features as STE), F_o, and F_r. As in the STE, F'_I and F'_M are passed through transformer encoders (here using 10 attention heads) to yield embeddings E_I and E_M. Then the object feature matrix F_o is encoded (via a transformer encoder without positional encoding, since object order is arbitrary) to produce an object embedding E_o. Likewise, the relation matrix F_r is encoded by a simple feed-forward network (no attention, since relations are unordered) to produce a relation embedding E_r.

Finally, these four embeddings are fused by element-wise addition:

$$E_{ORE} = E_I \oplus E_M \oplus E_o \oplus E_r.$$

This sum is the output of the ORE. Thus $E_{ORE} \in \mathbb{R}^{L \times d_{\text{model}}}$ encodes coarse video features (from E_I, E_M) enriched by object and relation semantics (E_o, E_r).

3.2.3 Backward Decoder (BD)

The Backward Decoder generates a caption in reverse order (right-to-left). It takes the STE output E_{STE} as its encoder memory; it does not use E_{ORE}, because object relations have no natural reverse-time ordering and could confuse the backward generation. The BD is implemented as a transformer decoder: at each step it attends to E_{STE} and generates one word from the end of sentence toward the beginning. Generation stops when the end marker $< S >$ is produced. If the generated reverse caption is $\overleftarrow{C} = [s_1, s_2, \ldots, s_L, < S >]$, then the final hidden state of the BD (after producing s_L) is denoted $\overleftarrow{D}_L$. We define the BD's context output as

$$\overleftarrow{D} = \overleftarrow{D}_L$$

This vector $\overleftarrow{D}$ summarizes the reverse caption's context and is passed on to the forward decoder.

3.2.4 Forward Decoder (FD)

The Forward Decoder generates the final caption left-to-right. It integrates two sources of context: the video encoding and the backward decoder's context. Concretely, the FD attends to E_{ORE} (via a standard encoder-decoder attention) so that object and relation features inform each predicted word. It also attends to the reverse caption context $\overleftarrow{D}$ via an extra cross-attention layer. In practice, at each step the FD takes as input all previously generated words and uses cross-attention over $\overleftarrow{D}$ so that it "sees" the whole reverse caption context. It similarly attends to E_{ORE} (which contains object/relation/video info). Generation ends at $< S >$, producing a forward caption $\overrightarrow{C} = [s_1, \ldots, s_T, < S >]$. Thus, the FD effectively conditions each word on both E_{ORE} and $\overleftarrow{D}$.

3.3 Training

BTKG is trained end-to-end with cross-entropy losses on both decoders. The training involves a two-stage process: first use the backward decoder to generate (pseudo) reverse captions, then use those in the forward decoder. Let D be the training set of videos with forward ground-truth captions. For each video V with forward caption $\overrightarrow{Y} = [y_1, \ldots, y_L]$, we first obtain a pseudo reverse caption (described below) and train the BD on that. The FD is then trained on the forward caption, conditioned on the BD's output. Denote the BD loss by L_{bd} and the FD loss by L_{fd} (both standard token-level cross-entropy). The total training loss is a weighted sum:

$$L = (1 - \lambda)\, L_{bd} + \lambda\, L_{fd},$$

where $\lambda \in [0, 1]$ balances the two parts.

3.3.1 Pseudo Reverse Captions

To avoid trivial information leakage, the reverse captions used for training are pseudo. All ground-truth forward captions for a video are reversed in order (word sequence flipped) to create candidate reverse captions of the same length. Then these reversed captions are randomly shuffled and paired with the original video, so the final training reverse caption is not exactly the true reverse of its forward caption. In other words, for training we do not feed the exact future ground-truth to the BD. This randomization mitigates the leakage issue and prevents the model from cheating by memorizing exact reversals.

3.3.2 Backward Decoder Loss

The BD is trained to minimize the negative log-likelihood of the pseudo reverse captions. Formally, if $(V, \overleftarrow{Y}) \in D$ where $\overleftarrow{Y}$ is a training reverse caption (of length T), then

$$L_{bd} = \sum_{(V,\overleftarrow{Y})\in D} \sum_{t=1}^{T} -\log p(y_t \mid V,\, y_1, \ldots, y_{t-1};\; \theta_{ste}, \theta_{bd}),$$

where y_t is the t-th token in $\overleftarrow{Y}$, and $\theta_{ste}, \theta_{bd}$ are the STE and BD parameters. This is the standard cross-entropy loss for the backward decoder.

3.3.3 Forward Decoder Loss

Similarly, the FD is trained by cross-entropy on the forward captions. Importantly, each forward token probability is conditioned on both the video and the reverse-context $\overleftarrow{D}$. The loss is:

$$L_{fd} = \sum_{(V,\overrightarrow{Y})\in D} \sum_{l=1}^{L} -\log p(y_l \mid V,\, y_1, \ldots, y_{l-1};\; \theta_{ste}, \theta_{bd}, \theta_{ore}, \theta_{fd}),$$

with $\theta_{fd}, \theta_{ore}$ the FD and ORE parameters. Though the form looks like a normal forward loss, recall that in the FD decoder architecture each step attends to $\overleftarrow{D}$ (the BD context), allowing the model to use “future” information encoded by the BD.

In summary, training optimizes $L = (1 - \lambda)L_{bd} + \lambda L_{fd}$ so that the BD learns to generate accurate reverse captions and the FD learns to generate accurate forward captions given both video features and the reverse-context. This bidirectional setup with pseudo-reverse captions is designed to reduce information leakage and fully exploit the video semantics captured by the Knowledge Graph encoder.

4 Keyframe-Based Image Captioning

Beyond the primary video-level architecture, we further investigate a complementary paradigm that treats video captioning as a sequence of keyframe-driven image-captioning tasks. In this variant, keyframes are first distilled from the clip and each is captioned by a image-captioning model, rather than processing the entire video sequence.

4.1 Keyframe Selection

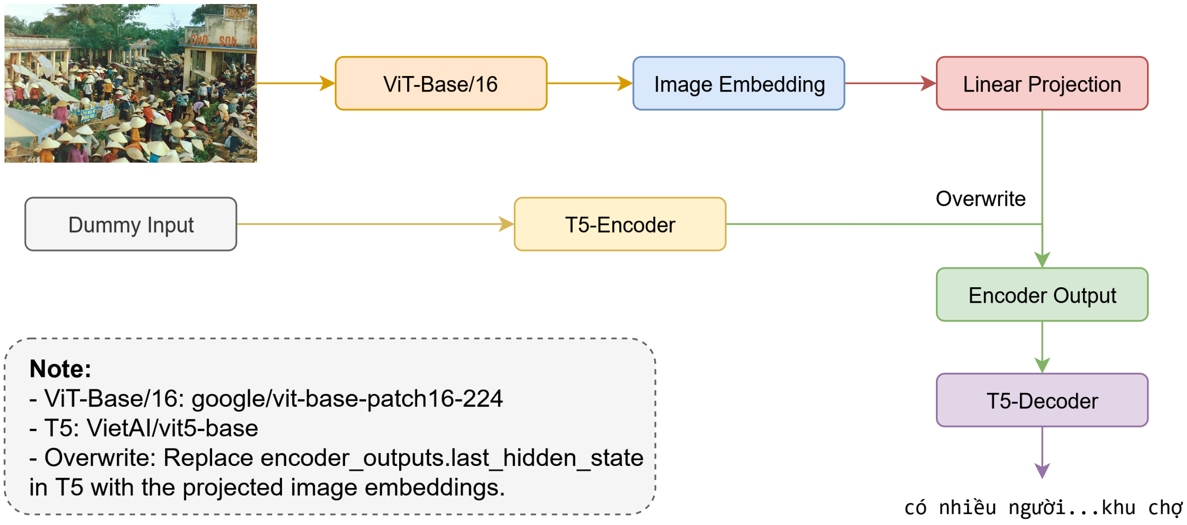
The keyframe selection procedure is designed to minimize redundancy and extract only those frames that represent significant visual changes within the video. This process involves the following steps:

1. Frame Sampling: Frames are initially sampled from the video at a rate of 1 frame per second (fps). This sampling rate substantially reduces the processing load while maintaining a linear runtime relative to the video duration, which is well-suited for short, single-activity videos typical of the MSVD dataset.
2. Feature Extraction via Visual Embedding: Each sampled frame is transformed into a high-dimensional feature vector using a Vision Transformer (google/vit-base-patch16-224-in21k [4]) model pre-trained on the ImageNet-21k dataset. ViT embeddings were selected for their demonstrated robustness to common visual artifacts such as motion blur and variations in illumination, rendering them a reliable foundation for change detection.
3. Similarity-Based Filtering: A frame is designated as a keyframe if its visual representation is sufficiently dissimilar from that of the preceding keyframe. This determination is made using a cosine similarity metric. A frame is added to the keyframe set if the cosine similarity score between its embedding and that of the last accepted keyframe is below a threshold of 0.95.

4.2 Image Captioning Model

4.2.1 Model Architecture

The image captioning model employs a sophisticated encoder-decoder architecture, integrating a pre-trained Vision Transformer (ViT) as the encoder and a pre-trained Transformer-based language model (T5) as the decoder.



Note:
- ViT-Base/16: google/vit-base-patch16-224
- T5: VietAI/vit5-base
- Overwrite: Replace encoder_outputs.last_hidden_state in T5 with the projected image embeddings.

Figure 2: ViT-T5 image captioning architecture.

The visual encoder is a ViT-Base/16 [5] model, pre-trained on the ImageNet-21k dataset. Its function is to process an input image and generate a sequence of patch embeddings, which serve as a rich, high-dimensional representation of the image's content.

The text decoder is based on the VietAI/vit5-base [6] model, a T5 variant specifically pre-trained for the Vietnamese language. To bridge the modality gap between the visual encoder and the text decoder, a dedicated interface mechanism was implemented. A linear projection layer maps the embedding dimension of the ViT's output features to the expected hidden dimension (d_{model}) of the T5 decoder.

During the forward pass, the projected visual features directly substitute the output of the T5's own text encoder. Consequently, the T5 decoder's cross-attention mechanism attends directly to this sequence of visual embeddings, enabling it to generate a textual description conditioned on the image content. For generation, the model utilizes a beam search with a beam width of 3 to produce fluent and coherent captions.

4.2.2 Training Procedure

The model was trained using a parameter-efficient fine-tuning (PEFT) strategy to leverage the knowledge from the pre-trained components while minimizing computational cost and the risk of catastrophic forgetting. During this phase, the weights of the entire ViT encoder and the T5 text encoder were frozen and did not receive updates. Only the parameters of the T5 decoder, the final language model head, and the intermediary linear projection layer were made trainable.

The model was optimized by minimizing the standard cross-entropy loss between the predicted token sequence and the ground-truth captions. We employed the AdamW optimizer with an initial learning rate of 5×10^{-5} and a weight decay of 1×10^{-4}. The learning rate was dynamically adjusted throughout training using a Cosine Annealing schedule with warm restarts.

Training was conducted for a maximum of 8 epochs. To prevent overfitting and select the best-performing model, an early stopping criterion was implemented. This mechanism monitored the validation loss at the end of each epoch and terminated the training process if no improvement was observed for 5 consecutive epochs. The model checkpoint that achieved the lowest validation loss was saved and used for all subsequent evaluations and inference tasks.

5 Experiments

5.1 Experimental Settings

5.1.1 Datasets

The **MSVD** dataset (Microsoft Research Video Description) was used as the evaluation benchmark. MSVD contains 1,970 short YouTube video clips (10-25 seconds each) with roughly 80,000 English sentences (about 40 captions per clip). These videos are split following standard practice: 1,200 clips for training, 100 for validation, and 670 for testing. (All captions are in English and describe a single-activity scene per clip, as in prior work.)

5.1.2 Feature extraction

For MSVD, we extract both spatial and motion features from the video clips. Video frames were sampled at **5 frames per second (fps)** and fed into a pretrained *Inception-ResNet-V2* network to obtain **image features**. Concurrently, videos were resampled at **25 fps** and divided into overlapping 64-frame segments (stride of 5 frames); these segments were processed by an *Inflated 3D ConvNet (I3D)* to compute **motion features**. The resulting image and motion feature vectors have dimensions **2048** and **1024**, respectively. We also encode each video's category label using a 300-dimensional GloVe word vector. For MSVD, 50 frames are uniformly sampled from each video's features, and all feature vectors (image, motion, and category) are projected to a 512-dimensional space.

5.1.3 Evaluation metrics

Performance on MSVD was measured with standard automatic captioning metrics: **BLEU**, **METEOR**, **CIDEr-D**, and **ROUGE-L**. BLEU and METEOR (originally developed for machine translation) quantify the quality of generated text by n-gram overlap with reference captions. CIDEr-D, explicitly designed for caption evaluation, measures consensus with human annotations. ROUGE-L (longest-common-subsequence based) was also used; it emphasizes recall in overlap between generated and reference sentences. These metrics together provide a comprehensive assessment of caption accuracy, relevance, and fluency.

5.1.4 Parameter settings

We detail the model's settings for MSVD experiments. The object detector (Mask R-CNN) used a confidence threshold of 0.7 and minimum box size of 224x224. A TransE knowledge-graph was built from 613 unique triples, with each relationship encoded by a 300-dimensional GloVe vector. The Transformer encoders and decoders use 4 layers, with 640-dimensional input embeddings, 512-dimensional model hidden states, 2048-dimensional feedforward layers, and 10 attention heads per layer. (An extended multi-head attention in the STE component uses 128 heads.) During training on MSVD, the balance weight λ between forward/backward decoding was set to 0.6 and the model was trained for 30 epochs. We used the Adam optimizer with a batch size of 32 and a learning rate of 1e-4 for MSVD training. Inference used beam search of size 5 with dropout 0.1. All experiments were run on NVIDIA Tesla P100 GPUs.

Reference

[1] Maosheng Zhong et al. "Bidirectional transformer with knowledge graph for video captioning". In: *Multimedia Tools and Applications* 83 (Dec. 2023), pp. 1–20. DOI: 10.1007/s11042-023-17822-4.

[2] Ashish Vaswani et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[3] Maosheng Zhong et al. “BiTransformer: augmenting semantic context in video captioning via bidirectional decoder”. In: *Mach. Vision Appl.* 33.5 (Sept. 2022). ISSN: 0932-8092. DOI: 10.1007/s00138-022-01329-3. URL: https://doi.org/10.1007/s00138-022-01329-3.

[4] Google. *ViT-Base-Patch16-224-In21k*. https://huggingface.co/google/vit-base-patch16-224-in21k. Accessed: 2025-06-08. 2024.

[5] Google. *ViT-Base-Patch16-224*. https://huggingface.co/google/vit-base-patch16-224. Accessed: 2025-04-08. 2024.

[6] Long Phan et al. “ViT5: Pretrained Text-to-Text Transformer for Vietnamese Language Generation”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*. Association for Computational Linguistics, 2022, pp. 136–142. URL: https://aclanthology.org/2022.naacl-srw.18.