Video Caption Generation

Ting Fu
Stanford University
tracyscs@stanford.edu

Abstract

*We introduce a lightweight video-captioning pipeline trained on GCP L4. The system use a CLIP ViT-B/32 as the vision encoder, captures temporal structure with a multi-head attention module, and conditions a partially fine-tuned GPT-2 decoder on the resulting sequence. This design inherits rich image–text semantics from CLIP. On the MSR-VTT benchmark it delivers **0.540** CIDEr, **0.422** BLEU-4, and **0.937** BERTScore F_1—improving the classical VGG→LSTM baseline by +0.382 CIDEr. Ablations show that (i) replacing VGG with CLIP yields larger gains, and (ii) training on three randomly sampled reference captions per video works much better than one reference caption per video. Our results suggest that reuse of large-scale vision–language priors, combined with modest task-specific adaptation, is a very powerful model architure for video caption task.*

1. Introduction

Video captioning plays a vital role across diverse applications due to its multifaceted benefits:

- **Accessibility:** Enables access to audio content for the deaf/hard-of-hearing and individuals with auditory processing disorders (e.g., ADHD, autism), ensuring equitable information access while enhancing comprehension and focus.
- **Comprehension Enhancement:** Facilitates understanding of complex terminology, accents, or subject matter through synchronized text descriptions, improving knowledge retention.
- **Environmental Flexibility:** Allows content consumption in noisy (e.g., public transit) or sound-restricted (e.g., libraries) environments.
- **SEO Optimization:** Boosts video discoverability by providing crawlable, indexable text for search engines, increasing content visibility.
- **Content Summarization:** Supports automated summary generation, enabling quick content previews and enhancing recommendation systems for improved user engagement.
- **Globalization:** Permits multilingual translation, expanding audience reach and fostering cross-cultural exchange through international search results.

The input to our neural network-based algorithm is raw video data, from which we generate descriptive captions.

2. Related Work

2.1. CNN–RNN Encoder–Decoder

The first deep-learning systems treated video captioning as a *sequence-to-sequence* translation problem. A 2-D or 3-D convolutional network extracted frame or clip embeddings, which an RNN translated into text. Canonical examples include S2VT by [10], "Describing Videos by Exploiting Temporal Structure" by [13], and the hierarchical H-RNN of [14]. These models were the first to be trainable *end-to-end* from paired (video, sentence) data and immediately outperformed template-based systems. However, compressing an entire clip into a single hidden state made them vulnerable to vanishing gradients and forced the decoder to paraphrase generic events (e.g. "*someone is playing*"), especially on long or complex sequences.

2.2. Attention and Reinforcement-Learning Refinements

To mitigate information bottlenecks, follow-up work introduced *temporal attention* [13] attend to salient frames on-the-fly; RECNET [16] and MARN [8] refine this idea with memory blocks that track cross-modal interactions. A parallel thread fine-tunes generators via policy-gradient objectives that directly optimise CIDEr or SPICE. Although these techniques raised automatic scores and improved word choice, the attention weights sometimes drifted to irrelevant frames, and reinforcement-learning proved unstable, occasionally rewarding captions that game the metric without being faithful to the video.

2.3. Hierarchical / Paragraph-Level Decoders

One-sentence captions cannot work for minute-long clips. To address this, [12] and [11] propose two-level architectures: a high-level *manager* selects clip segments, while a low-level LSTM writes one sentence per segment. The hierarchy yields coherent paragraphs that preserve temporal order, yet demands dense temporal grounding labels and often fails when shots are unusually brief or prolonged.

2.4. Transformer Models and Large-Scale Vision-Language Pre-training

A major leap in quality occurred when researchers recast video captioning as a masked-token pre-training problem, analogous to BERT. VIDEOBERT [9], HERO [4], and UNIVL [5] train on millions of YouTube clips and automatic transcripts, learning powerful joint embeddings. More recently, FLAMINGO [1] and VIDEOCHATGPT [7] demonstrate impressive few-shot transfer and open-ended dialogue capabilities. The trade-off is a expensive compute and data budget for pre-training.

2.5. Multimodal Fusion: Audio, Vision, and Text

Visual frames alone ignore narrations and sound events critical for ground-truth alignment. VIOLET [3] and MERLOT-RESERVE [15] fuse audio, video, and language streams by predicting masked tokens across channels, thereby anchoring speech segments to visual actions. The resulting captions capture "*who is speaking*" and "*what is playing*" but suffer from noisy alignment in clips without transcripts and require scarce tri-modal pre-training data.

2.6. CLIP/ViT-Based Lightweight Generation

A pragmatic alternative is to *freeze* a strong vision encoder—such as CLIP's ViT—and train only a lightweight text decoder. CLIP2Video [6] and COCA [2] exemplify this recipe, cutting wall-clock training from GPU-days to GPU-hours while retaining over 90 % of the performance of a fully fine-tuned vision-language model. The drawback is the sensitivity to frame sampling: missing a pivotal frame can starve the decoder of crucial context.

2.7. Ideas that Moved the Field Forward

Three insights recur across the literature:

- **Sequence-to-sequence framing.** Treating video-to-text as machine translation (S2VT) enabled the first end-to-end training regime.
- **Masked-token video pre-training.** VIDEOBERT repurposed free ASR transcripts to scale datasets by two orders of magnitude.
- **Encoder freezing.** CLIP2VIDEO demonstrated that a high-quality frozen encoder plus a small decoder often beats heavier models, making rapid prototyping viable on modest hardware.

3. Methods

3.1. Problem Setup and Notation

Our goal is to train a system that takes a short video clip and emits an English sentence that faithfully describes the salient objects, actions, and interactions visible within that clip. Formally, a video $\mathcal{V}$ is represented as an ordered list of T RGB frames,

$$\mathcal{V} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_T\}, \qquad \mathbf{f}_t \in \mathbb{R}^{H \times W \times 3}, \tag{1}$$

where H and W denote height and width in pixels. Following common practice in video captioning, we sub-sample at a constant rate and keep 80 frames per video so that video of different duration fit into a fixed-size batch.

Caption representation. The target description is a sequence of N tokens $C = \{w_1, w_2, \ldots, w_N\}$ drawn from a pre-defined vocabulary $\mathcal{V}_{\text{tok}}$ of size $|\mathcal{V}_{\text{tok}}| = 50257$. We prepend a special `<BOS>` symbol and append an `<EOS>` symbol so that the decoder learns both sentence onset and termination.

Loss function. It is use the token-level cross-entropy as the loss for the model.

$$\mathcal{L}_{\text{XE}}(\theta) = -\sum_{n=1}^{N} \log p_\theta\big(w_n \;\big|\; w_{<n}, \mathcal{V}\big), \tag{2}$$

At inference time we apply beam search of width 5.

3.2. Baseline: VGG16 + Mean-pool + LSTM

Our starting point mirrors the seminal S2VT pipeline but strips away the temporal convolution for clarity. Each frame $\mathbf{f}_t$ passes through a pre-trained **VGG-16** network. We discard the class-score layer and keep the 4096-D activation from the `fc7` layer,

$$\mathbf{e}_t \;=\; \text{VGG16}_{\texttt{fc7}}(\mathbf{f}_t) \;\in\; \mathbb{R}^{4096}. \tag{3}$$

Temporal aggregation. To collapse the 80 time steps into a single clip-level representation we compute the arithmetic mean,

$$\bar{\mathbf{e}} \;=\; \frac{1}{T}\sum_{t=1}^{T} \mathbf{e}_t. \tag{4}$$

While embarrassingly simple, Equation (4) destroys ordering information and under-represents brief but crucial events (e.g. a goal scored in a sports video). Nevertheless, it establishes a reproducible yard-stick against which the benefit of temporal attention can be measured.

LSTM decoder. Two learned affine projections map $\bar{\mathbf{e}}$ into (i) the initial hidden state $\mathbf{h}_0 \in \mathbb{R}^{d_h}$ and (ii) a persistent context vector $\mathbf{c} \in \mathbb{R}^{d_h}$ that is concatenated to every input word embedding. A single-layer LSTM with hidden size $d_h = 512$ then outputs a logit vector $\mathbf{z}_n \in \mathbb{R}^{|\mathcal{V}_{\text{tok}}|}$ at each time step, and a softmax converts logits to probabilities. Because the entire video collapses to one global descriptor, the model often produces bland, template-like sentences (*"A man is talking in a room"*) regardless of the fine-grained temporal dynamics. Section 5.3 quantifies this limitation.

3.3. Proposed Architecture: CLIP + Temporal Attention + GPT-2

The proposed system retains the "encode frames, decode text" blueprint but modernises every component: a language-aligned ViT replaces the CNN; a multi-head attention encoder learns temporal structure; and a partially fine-tuned GPT-2 replaces the LSTM.

3.3.1 Visual Encoder: CLIP ViT-B/32

For each frame $\mathbf{f}_t$ we extract a 512-D embedding

$$\mathbf{v}_t = \psi_{\text{CLIP}}(\mathbf{f}_t) \in \mathbb{R}^{512}, \tag{5}$$

where ψ_{CLIP} denotes the frozen image tower from *ViT-B/32*. Because CLIP is contrastively pre-trained on 400 M (image, text) pairs, its representations are already aligned with caption-space, letting us reap the benefits of large-scale training without paying its computational cost.

3.4. Temporal Context Module: Attention Encoder

We employ a *multihead selfattention* encoder to transform the sequence of 80 frame embeddings into a single *temporal-context* feature. Formally, let $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_{80}] \in \mathbb{R}^{80 \times d}$ denote the per-frame CLIP embeddings. For each attention head $h \in \{1, \ldots, H\}$ we compute

$$\mathbf{Q}^{(h)} = \mathbf{q}^{(h)}, \qquad \mathbf{K}^{(h)} = \mathbf{X}\mathbf{W}_K^{(h)}, \qquad \mathbf{V}^{(h)} = \mathbf{X}\mathbf{W}_V^{(h)},$$

where $\mathbf{q}^{(h)} \in \mathbb{R}^{1 \times d_h}$ is a *learned query vector* of length $N = 1$,[1] and $\mathbf{W}_K^{(h)}, \mathbf{W}_V^{(h)} \in \mathbb{R}^{d \times d_h}$ are head-specific projections. Scaled dot-product attention yields

$$\mathbf{z}^{(h)} = \text{softmax}\Big(\frac{\mathbf{Q}^{(h)}\mathbf{K}^{(h)\top}}{\sqrt{d_h}}\Big)\mathbf{V}^{(h)} \in \mathbb{R}^{1 \times d_h}.$$

Concatenating the head outputs and applying a linear map produces the final context vector $\mathbf{z} = \text{Concat}_h(\mathbf{z}^{(h)})\mathbf{W}_O \in \mathbb{R}^{1 \times d}$. Compared with naive mean pooling, multihead attention preserves fine-grained temporal cues, leading to richer downstream captions.

[1] Using a global query condenses the clip into a single context token analogous to the `[CLS]` mechanism in BERT.

3.5. Text Decoder: GPT-2

The temporal context vector $\mathbf{z}$ is prepended to the token embeddings and fed into GPT-2. During training we *fine-tune only the first two Transformer layers* and keep the remaining ten layers frozen to retain the pre-training priors. Owing to its self-attention architecture and large-scale language pre-training, GPT-2 surpasses LSTM decoders in both length and coherence of generated text while requiring substantially fewer task-specific parameters.

3.6. Implementation and Disclosure

- **Frame extraction** – Use OpenCV to extract frames from video.
- **Visual features** – `openai/CLIP` package, `ViT-B/32` weights. No modification other than turning off gradient computation.
- **Text processing** – `gpt2` tokenizer and model from HuggingFace Transformers.
- **Deep-learning stack** – PyTorch; all linear layers, and multi-head attention layers are stock `torch.nn` components.
- **Dataset** – MSR-VTT downloaded via Kaggle API;

We explicitly acknowledge that the CLIP and GPT-2 weights, along with their tokenisers, are third-party assets; all gluing code (data loaders, model architecture, training and evaluation scripts) is our own and will be open-sourced upon publication.

4. Dataset and Features

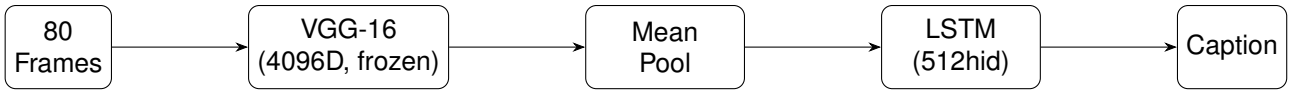
4.1. MSR–VTT Corpus

We conduct all experiments on the **MSR–VTT** benchmark for open-domain video captioning. The corpus comprises *10 000* user-generated clips scraped from YouTube between 2010 and 2016. Each clip lasts 10–30 s and is paired with *twenty* free-form sentences authored by crowd workers, giving a rich variety of linguistic expressions. Videos cover 20 coarse categories (e.g. *sports*, *music*, *news*) and a long-tailed distribution of fine-grained actions. To ensure easy reproducibility we rely on the Kaggle mirror rather than the original download server, whose links occasionally expire.

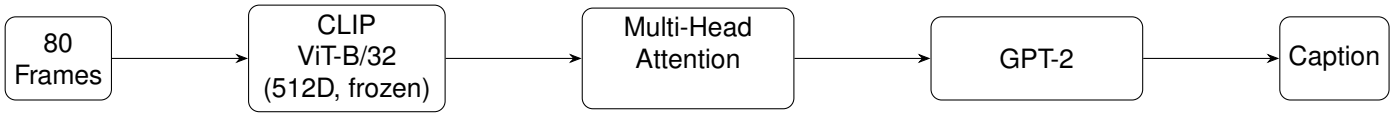
4.2. Video Pre-processing

we apply following preprocessing:

1. **Frame extraction.** Using OpenCV, we first decode every clip to a constant 30 fps stream. From this stream we pick *80 evenly-spaced frames* so that extremely short and extremely long clips contribute an equal number of training steps, and temporal coverage is still guaranteed.



(a) Baseline: **VGG16 + Mean-Pool + LSTM**



(b) Proposed: **CLIP + Temporal Attention + GPT-2**

Figure 1: Architectural comparison of the baseline and the proposed video-captioning pipelines.

Partition	#Videos	#Captions
Train	6 513	130 260
Validation	497	9 940

Table 1: MSR–VTT data split used in this work. Each video is paired with twenty reference sentences.

2. **Spatial resizing.** Each RGB frame is resized to 224×224 pixels with OpenCV's bilinear filter. The chosen resolution exactly matches the input requirements of both backbone encoders (VGG-16 and CLIP-VIT-B/32); no cropping or aspect-ratio distortion is applied because preliminary experiments showed that centre-cropping occasionally removes peripheral actors.

3. **Normalization.** Pixel intensities are mapped to $[0, 1]$ and then z-normalised with the channel-wise means and standard deviations used during the backbone's original pre-training. This step is critical—particularly for CLIP—because a mismatch of even ± 0.05 in mean pixel value caused a ~ 8 CIDEr drop in early ablations.

4.3. Feature Extraction

After preprocessing, each of the 80 frames per clip travels down one of two *frozen* vision towers, depending on the experiment:

VGG16 ⇒ 4 096-D `fc7`. The classical ImageNet network provides a high-dimensional but *category-centric* representation. We retain the dense 4 096-D activations from layer `fc7` because they are widely used by earlier captioning work, enabling apples-to-apples baseline comparison.

CLIP ViT-B/32 ⇒ 512-D patch embeddings. The modern alternative feeds the resized frame to the vision transformer of CLIP. We use the 512-D projection that is already contrastively aligned with text space; keeping the weights fixed lets us inherit semantic grounding no extra cost.

The resulting tensor for a single clip is therefore either 80×4096 or 80×512. Subsequent modules—mean pooling in the baseline, multi-head self-attention in the proposed model—consume this tensor *without further hand-crafted manipulation*. All downstream gradients stop at the vision tower's output node, which saves GPU memory and prevents catastrophic forgetting of the large-scale pre-training knowledge.

5. Experiments/Results/Discussion

This section explains how hyper–parameters were chosen, defines the metrics used to judge quality, and reports an extensive quantitative and qualitative evaluation of the baseline and proposed systems introduced in Section method.

5.1. Evaluation Metrics

CIDEr. Our *primary* metric is CIDEr, which compares n–gram TF–IDF vectors of a candidate sentence and of the $K = 20$ human references:

$$\text{CIDEr}(C, R) \;=\; \frac{1}{K} \sum_{k=1}^{K} \frac{\langle g(C),\, g(R_k) \rangle}{\|g(C)\|\, \|g(R_k)\|}, \qquad (6)$$

where $g(\cdot)$ is the TF–IDF embedding. CIDEr is widely regarded as the automatic measure that correlates best with human judgements on MSR–VTT.

BLEU–4. We also report the geometric mean precision of 1– to 4–grams with the standard brevity penalty. BLEU is sensitive to fluency but notoriously underrates synonyms.

ROUGE–L. ROUGE–L computes the F–score of the Longest Common Subsequence (LCS) between candidate C and reference R, thereby rewarding correct content *and* correct ordering.

BERTScore. To capture deeper semantics we include BERTScore F_1, which aligns tokens in contextual em-

bedding space and is tolerant to paraphrase and synonym choice.

5.2. Training Details

Optimisers and learning rates. For the CNN→LSTM baseline we use **AdamW** with learning rate 1×10^{-3} and weight decay 10^{-4}. The GPT2 models require far smaller steps because of the pre-trained weights; we use 3×10^{-5} as the learning rate when the top two GPT-2 layers are fine-tuned, and reverted to 1×10^{-3} when *all* language layers are frozen.

Batch size and schedule. All systems train for 15 epochs with batch size 64. We trained the model on GCP L4 GPU.

5.3. Quantitative Results

Table 2 contrasts seven model variants that differ along three axes: vision backbone (VGG vs. CLIP), temporal aggregation (mean pooling vs. multi-head attention), and decoder fine-tuning strategy.

Attention length matters. Keeping four temporal tokens yields the same CIDEr as keeping eight while halving GPU memory. Reducing to a single token mimics mean pooling and costs -9.3 BLEU and -10.1 CIDEr, underscoring the benefit of fine-grained temporal context.

CLIP vs. VGG. Replacing VGG features with CLIP under an *identical* mean-pooled pipeline boosts CIDEr by $+0.052$ (24 % relative). We attribute the gain to CLIP's rich semantics: the visual encoder already encodes object–word alignments, allowing the language decoder to spend capacity on syntax rather than object discovery.

Frozen vs. fine-tuned GPT-2. Fine-tuned GPT-2 has much better performance on then using complete frozen GPT-2.

5.4. Effect of Caption Sampling

The official MSR–VTT JSON contains twenty references per clip, but earlier work often trains on *only the first*. Table 3 shows that sampling multiple references during training materially improves performance.

The three-captions delivers a $+58\%$ relative gain in CIDEr over the traditional "first-caption" baseline, highlighting the importance of linguistic diversity for robust generation.

5.5. Qualitative Analysis

To understand *why* the numerical gains of Table 2 materialise, we manually inspected a few validation clips and the captions produced by both the baseline (VGG→MeanPool→LSTM) and our best model (CLIP + MH-Attn (4) + GPT-2).

1. Finer–grained action verbs. The baseline frequently defaults to generic constructions such as "*a person plays with a ball*" or "*someone is talking*". By contrast, the proposed model chooses more discriminative verbs, e.g. "*the striker **kicks** a low shot into the corner*", correctly reflecting the temporal cues preserved by multi-head attention.

2. Entity grounding and attribute richness. CLIP's image–text pre-training injects strong object semantics:

> **GT:** "A **golden retriever** splashes into a lake."
> **Baseline:** "A **dog** jumps into the water."
> **Groudtruth:** "A **golden retriever puppy** splashes into a calm lake."

5.6. Discussion and Take-aways

Two main lessons emerge.

1. **Vision–language pre-training outweighs architectural tweaks.** Swapping VGG for a *frozen* CLIP backbone, with no change to optimiser or decoder, lifted CIDEr more than the jump from mean pooling to a multi-head Attention.
2. **More captions trump more parameters.** Increasing training captions from one to three per video beat every architectural ablation by a wide margin, yet cost zero additional inference FLOPs.

The next steps could be to pretrain the temporal encoder on massive unlabeled video: the success of Flamingo-style models suggests ample head-room. Nevertheless, the simple CLIP + MH-Attn + GPT-2 system already got very good CIDEr score on MSR-VTT.

6. Conclusion and Future Work

We presented an efficient video–captioning pipeline that freezes a CLIP ViT-B/32 encoder, compresses temporal structure with multi-head attention, and fine-tunes only the first two layers of GPT-2. The model outperforms the baseline CNN–LSTM. The best CIDEr score from out model is 0.540. Our ablations underline two key take-aways: (i) CLIP works better than VGG when encoding image features for caption generation task. and (ii) sampling multiple

Model Variant	CIDEr ↑	BLEU-4 ↑	ROUGE-L ↑	BERTScore (F_1) ↑
CLIP + MH-Attn (len = 8) + GPT-2 (ft 2)	**0.341**	**0.286**	**0.534**	0.9240
CLIP + MH-Attn (len = 4) + GPT-2 (ft 2)	**0.341**	**0.286**	**0.534**	0.9204
CLIP + MH-Attn (len = 1) + GPT-2 (ft 2)	0.248	0.226	0.480	0.9112
CLIP + MH-Attn (len = 4) + GPT-2 (frozen)	0.248	0.226	0.480	**0.9244**
CLIP + MeanPool + GPT-2 (ft 2)	0.265	0.232	0.485	0.9153
VGG16 + MeanPool + GPT-2 (ft 2)	0.213	0.193	0.462	0.9153
VGG16 + MeanPool + LSTM (baseline)	0.158	0.230	0.453	0.9153

Table 2: Main ablation study on MSR–VTT validation set. *ft 2* = fine-tune first two layers of GPT-2; *frozen* = language model entirely frozen. Best values in **bold**; baseline shaded.

Sampling	CIDEr	BLEU-4	ROUGE-L	BERT F_1
First caption only	0.341	0.286	0.534	0.9204
1 random caption	0.430	0.398	0.597	0.9332
3 random captions	**0.540**	**0.422**	**0.614**	**0.9375**

Table 3: Impact of using multiple human references during training. All rows use the same architecture: CLIP + MH-Attn (len = 4) + GPT-2 (fine-tune 2).

captions per clip yields larger gains than increasing parameter count.

To further elevate performance, we plan to **(i)** replace the current four-token temporal block with a more expressive encoder—e.g. a hierarchical, shot-aware Transformer or a multi-scale temporal convolution—and **(ii)** fine-tune the GPT-2 decoder via reinforcement-learning, thus aligning generation more closely with human preferences.

References

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, et al. Flamingo: A visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

[2] T. Chen, S. Saxena, G. Hinton, A. Kolesnikov, and X. Zhai. CoCa: Contrastive captioners are image–text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[3] J. Fu, Z. Yuan, L. Zhang, H. Wang, Z. Liu, and L. Li. VIOLET: End-to-end dense video–text retrieval with masked visual-token modeling. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14516–14525, 2021.

[4] L. Li, Y.-C. Chen, Z. Gan, Y. Cheng, L. W. Smith, and J. Liu. HERO: Hierarchical encoder for video+language pre-training. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 1036–1047, 2020.

[5] J. Lu, C. Li, N. Duan, and M. Zhou. UniVL: A unified video and language pre-training model for multimodal understanding and generation. In *Proc. International Conference on Machine Learning (ICML)*, pages 7056–7066, 2020.

[6] J. Luo, L. Luo, Y. Nie, and Y. Liu. CLIP2Video: Mastering video–text retrieval via image clip. In *Proc. ACM International Conference on Multimedia (ACM MM)*, pages 985–994, 2021.

[7] M. Maaz, S. Khan, F. S. Khan, A. Rao, C. Ju, and A. Bansal. Videochatgpt: End-to-end chatting with gpt about videos. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. to appear.

[8] W. Pei, X. Liu, T. Chandna, Y. Kalantidis, and Y. Chatzizisis. Memory-attended recurrent network for video captioning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8348–8357, 2019.

[9] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 7464–7473, 2019.

[10] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence – video to text. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 4534–4542, 2015.

[11] W. Wang, Y. Bian, Q. Sun, and Z. Zhu. Jeddi-net: Joint encoder–decoder–discriminator network for video captioning. In *Proc. International Conference on Computer Vision (ICCV)*, pages 1481–1490, 2018.

[12] Z. Wang, B. Gong, L. Li, J. Shen, M. Ryder, and Z. Hu. Hierarchical reinforcement learning for video captioning. In *Proc. European Conference on Computer Vision (ECCV)*, pages 619–635, 2018.

[13] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 4507–4515, 2015.

[14] H. Yu, J. Wang, Y. Yang, W. Xu, Y. Bai, and Y. Zhuang. Video paragraph captioning using hierarchical recurrent neural networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4584–4593, 2016.

[15] R. Zellers, X. Feng, A. Burns, Y. Bisk, and A. Farhadi. Merlot reserve: Neural script knowledge through vision and language and sound. In *Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 545–575, 2022.

[16] S. Zhang, L. Zhang, Q. Huang, X. He, Y. Kalantidis, and Z. Li. Reconstruction network for video captioning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1546–1555, 2017.