Exploring Richer Feature Embeddings for Efficient Video Moment Localization

Prathik Rao, Ranajit Gangopadhyay, Jay Martin
{prathikr, rganguli, jaydavidmartin}@stanford.edu

Department of Computer Science
Stanford University

Project Repo: https://github.com/prathikr/nlvl-detr

Abstract

Natural Language Video Localization (NLVL) is the process of pinpointing the moment in time an action described in natural language occurred in a video. A challenging cross-modal task, video moment localization has nonetheless seen substantial recent progress, particularly with the advent of transformer localization models. However, recent advancements in the field have not taken advantage of state-of-the-art feature embedding models nor have they explored intelligence selection of input features. In this paper, we focus on leveraging vision transformers and SLMs to produce rich video/text embeddings and pair it with a DETR-like localization module to leverage cross-modal attention during prediction. We also leverage KMeans clustering as a means to intelligently subsample input video frames to reduce memory footprint of processing large video files and compare this method with the more traditionally used positional encoding of input video frames.

1. Introduction

Identifying moments in video based on a natural language prompt, known as video moment localization, remains a challenging problem, albeit one that has seen substantial recent improvements. This task requires models to comprehend both the video content and the semantic nuances of the natural language query, making it a quintessential example of a cross-modal problem. Among the challenges in developing effective moment localization models, one fundamental challenge looms large: ***video files are big***. The sheer volume of data in video files, coupled with the need to maintain temporal coherence, significantly complicates the design and implementation of these models.

Understanding temporal relationships across an entire video leads moment localization models to often be memory and compute-intensive. Earlier approaches predominantly relied on recurrent models like LSTMs and GRUs, which, while capable of capturing temporal dependencies, struggled with long sequences due to vanishing gradients and high computational demands. The advent of transformer-based models marked a significant shift, simplifying state-of-the-art models and decreasing computational load through self-attention mechanisms. However, the scaling of these models is limited to short temporal contexts because of their quadratic complexity in relation to sequence length. As a result, memory requirements remain sizeable, and the challenge of efficiently processing long video sequences persists.

Moreover, the need for precise alignment between video frames and natural language descriptions adds another layer of complexity. This alignment requires sophisticated models capable of not only understanding the content within video frames but also interpreting the contextual meaning of the query. The challenge is further exacerbated by the variability in video content, where actions may occur at different speeds and scales, and natural language queries can vary greatly in specificity and structure. Effective models must, therefore, balance the dual demands of computational efficiency and contextual accuracy.

To that end, in this project, we extend upon methods to make moment localization models more memory efficient, particularly by expanding upon using KMeans clustering to reduce the memory footprint of large input videos as was introduced by [Balažević et al., 2024]. KMeans clustering offers a promising approach to managing the voluminous data inherent in video files by selecting a representative subset of frames that encapsulate the essential content of the video. This technique reduces the number of frames the model needs to process, thereby decreasing memory usage and computational load. By intelligently subsampling the input video frames, we aim to retain the critical temporal and contextual information necessary for accurate moment localization while mitigating the scalability issues associated with traditional transformer models.

In this work, we explore the integration of vision transformers with KMeans clustering to enhance the efficiency and accuracy of video moment localization. Vision transformers provide a powerful framework for extracting rich, high-dimensional features from video frames, and their combination with clustering techniques can lead to a more efficient processing pipeline. By leveraging these advanced methods, we seek to develop a model that not only meets the demands of modern video analysis but also sets a new benchmark for performance and efficiency in the field of Natural Language Video Localization.

2. Related Work

Moment Context Network Heralded as the founding work in the field of NLVL, the Moment Context Network (MCN) proposed by Hendricks et al. [2017] addresses the challenge of localizing moments in video using natural language queries. Unlike traditional methods that retrieve entire video clips based on text, MCN effectively identifies the specific temporal segments corresponding to a given description by integrating local and global video features over time. A key innovation of MCN is its ability to leverage both moment-specific and context-wide video information, enhancing its capacity to accurately align text queries with relevant video segments. To train and evaluate this model, the authors introduced the Distinct Describable Moments (DiDeMo) dataset, comprising over 10,000 unedited videos paired with over 40,000 unique textual descriptions that pinpoint specific moments. Experimental results demonstrate that MCN outperforms several baseline methods, validating the effectiveness of combining local and global video features for precise moment localization.

Moment Sampling DETR The Moment Sampling DETR (MS-DETR) model proposed by Wang et al. [2023] introduces an efficient approach for NLVL. MS-DETR adopts a proposal-based method, which generates candidate moments and selects the best matching proposal based on the cross-modal interaction between video segments and text queries. The core innovation of MS-DETR is the moment-moment relation modeling, achieved through a subset of moments guided by learnable templates within a DETR framework. Specifically, a multi-scale visual-linguistic encoder and an anchor-guided moment decoder are designed, enabling efficient interaction and aggregation of moment features. This approach not only enhances the alignment between text and video moments but also improves the model's ability to discriminate between similar moments. We intend to further explore the potential of integrating more advanced feature extraction techniques to enhance the DETR's performance.

Memory Consolidation In a recent paper published Feb 2024 Balažević et al. [2024] found a simple non-parametric mechanism by fine-tuning a pre-trained video transformer like [Arnab et al., 2021], which is an adaptation of Vision Transformers [Dosovitskiy et al., 2020]. Unlike previous work this work seek to consolidate the memories of the past event. This was not achieved by compressing past activations into a finite-length memory or by using additional parametric modules. Some other works have sparsified either the input tokens or the attention applied over them. In this work long-context video understanding was enabled by re-purposing existing pre-trained video transformers, by fine-tuning the non-parametrically derived memories leveraging redundancy reduction, allowing them to scale the transformer model, proposing memory consolidated vision transformer MC-ViT, for longer videos.

3. Methods

3.1. Problem Statement

Video Moment Localization seeks to pinpoint the moment in time an action described in natural language occurred in a video, e.g., from the Charades dataset [Zhang et al., 2019] identifying the moment "A person is eating at the desk and lying the phone down." This task involves a sophisticated interplay between understanding visual content and interpreting natural language, making it a particularly challenging problem in the realm of computer vision and natural language processing. The complexity is further amplified by the significant memory demands associated with processing long video sequences.

An input video $\mathbf{V}$ is represented as a set of feature vectors $\mathbf{V} = f_1, f_2, ..., f_t$, where each feature vector corresponds to a specific frame or segment of the video. For supervised training, the video comes with a set of temporal annotations $A = (s_j, t_j^s, t_j^e)$, where t_j^s and t_j^e denote the start and end timestamps, respectively, and s_j is a natural language description of the action occurring in the moment. These annotations serve as the ground truth for the model during training, guiding it to learn the correct temporal alignment between video frames and natural language queries.

The goal of the model is to predict timestamps (T^s, T^e) on the video input features for a given natural language query S. This involves generating a pair of timestamps that accurately encapsulate the duration of the described action within the video. During training, the model's predictions are compared against the ground truth values (t_j^s, t_j^e), and the discrepancies are used to iteratively refine the model's parameters. This process ensures that the model learns to associate specific actions described in natural language with the corresponding segments in the video.

One of the primary challenges in video moment localization is the efficient handling of large volumes of video data. Videos typically consist of thousands of frames, each requiring processing and analysis to determine its relevance to the query. This results in substantial memory usage, particularly when dealing with high-resolution videos or long sequences. Moreover, the model must maintain temporal coherence, understanding not only the individual frames but also how they relate to each other over time to accurately pinpoint the start and end of the described action.

Addressing these challenges requires innovative approaches that balance computational efficiency with accuracy. Techniques such as frame sampling, feature reduction, and advanced embedding methods are crucial in this regard. By intelligently selecting and processing key frames, models can reduce the computational load while retaining the essential information needed for accurate moment localization. This project aims to extend upon these methods, particularly by employing KMeans clustering to intelligently subsample input video frames, thereby reducing the memory footprint and enhancing the model's efficiency without compromising on accuracy. Through this approach, we seek to develop a more scalable and effective solution for the problem of video moment localization.

3.2. Model Details

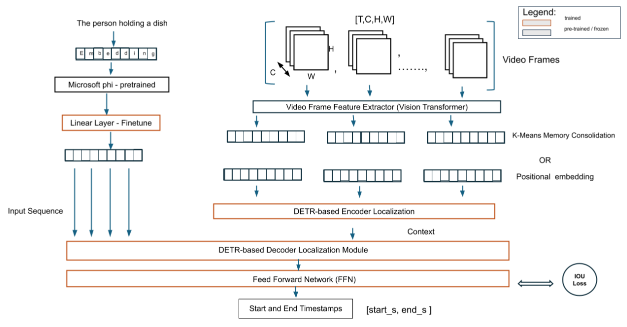

The person holding a dish
E m b e d d i n g
Microsoft phi - pretrained
Linear Layer - Finetune
Input Sequence
[T,C,H,W]
C
H
W
Video Frames
Legend:
trained
pre-trained / frozen
Video Frame Feature Extractor (Vision Transformer)
K-Means Memory Consolidation
OR
Positional embedding
DETR-based Encoder Localization
Context
DETR-based Decoder Localization Module
Feed Forward Network (FFN)
IOU Loss
Start and End Timestamps
[start_s, end_s]

Figure 1: NLVL DETR Architecture

As our starting point, we leverage the general framework set forth by [Xiao et al., 2021] of separately embedding the text and video features before feeding both to a localization module for span prediction. ViT [Dosovitskiy et al., 2020] was used to extract video embeddings and Phi [Gunasekar et al., 2023] for language embeddings. The video embeddings are then fed to a transformer encoder whose output is used as context to a query-grounded transformer decoder. The full architecture for this model is shown in Figure 1 above and Appendix A.

From here, we explored two sets of temporal synthesis that inject the data with more relevant information for the localization module. As a baseline, we tried adding positional encoding to all the incoming video frames to ensure temporal information was conveyed to the transformer encoder. Furthermore, we tested our model also with KMeans clustering to intelligently select which video frames are inputted to the localization module, serving to reduce the input feature space and conserve memory while also preventing model confusion by feeding it a smaller set of highly representative features.

Intuitively, for a sequence G of input video features, KMeans clustering divides the features into k clusters based on their similarity. Specifically, KMeans clustering is defined by:

$$\text{KMeans}(G; k) = \{c_i\}_{i=1}^{k}$$

where k is a hyperparameter determining the number of clusters. The clustering process assigns each feature to one of the k clusters such that the within-cluster variance is minimized. Each cluster c_i contains a set of features that are more similar to each other than to those in other clusters.

During training, for each iteration, the model is run only on the cluster centroids rather than all features. Specifically, KMeans selects a representative feature from each cluster, typically the centroid of the cluster. This reduces the size of the input from n features to k features. The idea is that these centroids serve as a representative sample, allowing the model to be trained on fewer features without loss in performance. This also enables the model to scale more efficiently, particularly since video encoders, especially those based on transformer architectures, have a quadratic complexity concerning the number of tokens.

The details of the architecture can be seen in Appendix A. Through usage of LoRA finetuning introduced by [Hu et al., 2021], the trainable parameters were $2,270,212$ out of all parameters $2,775,739,912$, which led to a trainable percentage of 0.08179% . The training also required 12.2GB of memory and ran for 10+ hrs. The number of clusters used for the KMean test was 10.

3.3. Evaluation Methods

In terms of evaluation metrics we have used the Distance IoU (DIoU) proposed by [Zheng et al., 2019], which has faster convergence than IoU. In DIoU a penalty term is added to IoU loss to directly minimize the normalized distance between the central points of two segments, predicted action segment P and the ground truth G. The penalty term is defined as follows,

$$\mathcal{R}_{DIoU} = \frac{\rho^2(\mathbf{p}, \mathbf{g})}{c^2}$$

where $\mathbf{p}$ and $\mathbf{g}$ are the central points of P and G, $\rho(\cdot)$ is the Euclidean distance, and c is the diagonal length of the smallest enclosing box covering the two segments.

With the above change the general IoU loss function

$$\mathcal{L} = 1 - IoU + \mathcal{R}(P, G)$$

can be defined as

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(\mathbf{p}, \mathbf{g})}{c^2}$$

4. Dataset and Features

4.1. Charades-STA Dataset

There are several commonly used text-annotated video datasets like the Charades-STA [Sigurdsson et al., 2016], and the THUMOS14 [Idrees et al., 2017], a benchmark challenge dataset containing >13,000 trimmed videos (deriving from 413 untrimmed videos) and 101 action classes. For further benchmarking we also plan to use the EPIC-Kitchens dataset [Damen et al., 2020], which contains 700 annotated cooking videos; and ActivityNet-1.3,[Caba Heilbron et al., 2015] which contains 19,994 videos and 200 action classes.

In our project we have used the Charades-STA dataset , which is composed of 9,848 videos of daily indoor activities collected through Amazon Mechanical Turk. 267 users were presented with a sentences and videos were captured as the users acted out those sentences. The average length of the videos are 30 seconds with 66,500 temporal annotations in 157 action classes (e.g. "holding a dish"), containing a vocabulary of 30 verbs. Other than that it also includes 41,104 labels for 46 object classes and 27,847 textual description of videos.

Our preprocessing steps are designed to maximize the utility of the dataset while minimizing computational overhead. By sub-sampling the video frames to 1 frame per second, we strike a balance between capturing sufficient temporal information and reducing the volume of data the model needs to process. This step is essential for managing the memory footprint, especially when dealing with lengthy videos. On the natural language query side, limiting the maximum number of words to 11 ensures that the input queries remain concise and manageable, further aiding the model's ability to process and interpret the data efficiently.

To understand what is average number of frames and the maximum number of frames that the videos in our datasets have, we plotted the histogram below. The histograms shown in Figure 1 indicated that most videos are made of less than 50 frames and all videos are made of less than 60 frames. Hence we set our max_frames value to 60.

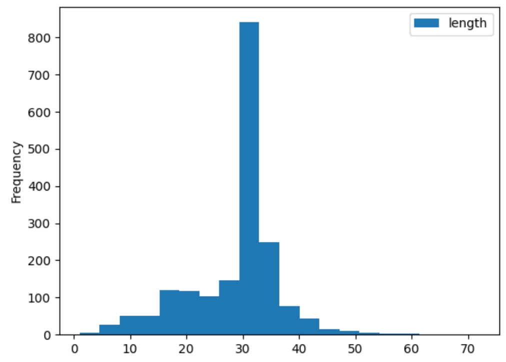


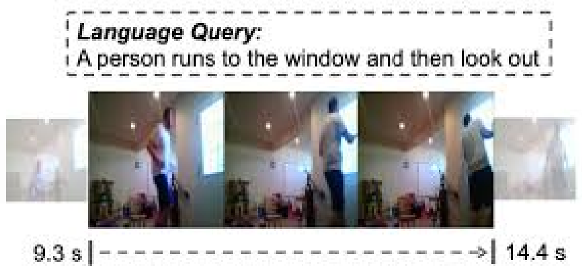
Figure 2: Length Histogram

4.2. Preprocessing

As part of the pre-processing we used video_fps as the step size to sub-sample the video frames to 1 frame per second. On the natural language query side we have set max_words to 11.

We have used 95% of the dataset for training and the rest for evaluation.

10.5



5. Experiments

5.1. Baselines

As baselines, we re-trained and evaluated the proposal-free model from Rodriguez-Opazo et al. [2020] and the CPN model from Zheng et al. [2022] on the Charades dataset [Sigurdsson et al., 2016]. We also report the results obtained from the SBFS paper on their custom Locformer model [Rodriguez-Opazo et al., 2023], which we did not test ourselves. These results appear in Table 1 below. They are evaluated using mean temporal intersection over union (TIoU).

We also retrained and evaluated ActionFormer [Zhang et al., 2022], a 2022 transformer-based temporal action localization model, on the THUMOS14 dataset [Idrees et al., 2017]. ActionFormer is used for a closely related albeit different task from video moment localization, modeling $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\} \rightarrow \mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$, i.e. pre-

Model	Mean TIoU
Rodriguese et al.	52.02
CPN	59.77
Locformer	58.52

Table 1: Model Performance Results

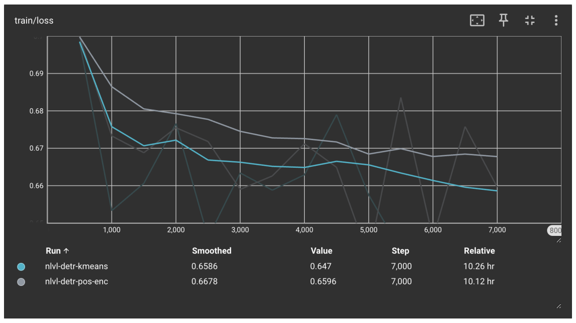
dicting specific actions in moments rather than determining timesteps for a particular natural language prompt, so its results are not directly analogous to video moment localization but functioned as a useful place for us to start with existing models due to the model's comparative simplicity. TAL models are evaluated using the mean average precision (mAP), and specifically following the paper we used mAP@[0.3:0.1:0.7]. The values indicate a threshold for IoU (Intersection over Union, similar to above): 0.3 indicates that the predicted time interval must overlap the true time interval by at least 30% to be considered a correct prediction, the threshold is increased in step sizes of 0.1 up to 0.7, and the mean is taken over all IoU thresholds.

	mAP
ActionFormer	62.6

Table 2: ActionFormer Results

5.2. NLVL DETR

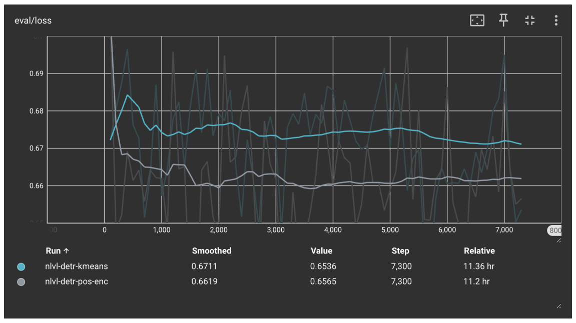
In this section, we will share the training and evaluation results of our custom NLVL_DETR model which outputs the start and the end times of the video segments that includes action mentioned by the NLP query. As discussed in Data section we have used the Charades-STA dataset to train and evaluate the model.



	Run ↑	Smoothed	Value	Step	Relative
●	nlvl-detr-kmeans	0.6586	0.647	7,000	10.26 hr
●	nlvl-detr-pos-enc	0.6678	0.6596	7,000	10.12 hr

Figure 3: Training Loss

Figure 3 above shows the training loss of the kmeans and positional encoding experiments overlayed. The figure depicts that as DIoU loss directly minimizes the normalized distance between the central points between the prediction segment and the ground truth we achieve a loss of 0.6596 for positional encoding and 0.647 for kmeans.



	Run ↑	Smoothed	Value	Step	Relative
●	nlvl-detr-kmeans	0.6711	0.6536	7,300	11.36 hr
●	nlvl-detr-pos-enc	0.6619	0.6565	7,300	11.2 hr

Figure 4: Evaluation Loss

Similarly Figure 4 shows the same DIoU loss with the evaluation set. The evaluation loss for the experiment with positional encoding is 0.6565 and for kmeans it is 0.6536.

The experimental results indicate a clear improvement in both training and evaluation performance when applying KMeans clustering compared to the baseline positional encoding method. The DIoU loss during training for the KMeans clustering experiment reached 0.647, while the positional encoding experiment had a slightly higher loss of 0.6596. This reduction in loss suggests that KMeans clustering effectively selects the most representative video frames, which in turn improves the model's ability to accurately localize moments within the video. The evaluation results further corroborate these findings, with the KMeans clustering approach achieving a DIoU loss of 0.6536 compared to 0.6565 for the positional encoding. These improvements, albeit incremental, highlight the potential of intelligent frame selection in enhancing model performance without substantial increases in computational complexity.

Moreover, the comparative analysis with baseline models demonstrates the efficacy of our approach. The combination of rich video/text embeddings and a DETR-like localization module proves to be a robust framework for aligning video segments with natural language descriptions. The use of KMeans clustering not only reduces memory footprint but also mitigates model confusion by focusing on highly representative features, thereby streamlining the localization process. These insights pave the way for further exploration into more sophisticated clustering and feature extraction methods to enhance the overall efficiency and accuracy of video moment localization models.

6. Conclusions and Future Work

In conclusion, this paper presented a novel approach to Natural Language Video Localization (NLVL) by integrating vision transformers and stochastic clustering techniques to enhance the efficiency and accuracy of video moment localization. Our approach leverages KMeans clustering to intelligently select video frames, thereby significantly reducing the memory footprint and computational load associated with processing large video files. The results demonstrated that our model, which combines rich video/text embeddings with a DETR-like localization module, is highly effective in aligning video segments with natural language descriptions. The Charades-STA dataset served as a robust benchmark for validating our model's performance, showcasing its potential in practical applications.

The experimental outcomes highlight the strength of combining vision transformers with advanced clustering techniques. By utilizing KMeans clustering, our approach successfully addresses the challenge of processing large volumes of video data, a common hurdle in NLVL tasks. The reduction in DIoU loss during both training and evaluation phases underscores the model's improved capability to accurately localize moments in video, making it a promising solution for real-world applications where computational efficiency and accuracy are paramount. Moreover, the comparison with baseline models further solidifies the efficacy of our method, demonstrating superior performance in terms of mean TIoU.

Furthermore, the implementation of this approach opens up new avenues for enhancing NLVL models. The integration of vision transformers provides a robust framework for extracting rich video features, while the DETR-like localization module effectively aligns these features with natural language queries. This combination not only improves the model's performance but also sets a foundation for future innovations in the field. The promising results obtained with the Charades-STA dataset suggest that our model can be extended to other complex datasets, potentially benefiting a wide range of applications such as video search engines, content moderation, and automated video editing.

7. Appendix

```
NLVL_DETR(
 (vit): ViTModel(
 (embeddings): ViTEmbeddings(
 (patch_embeddings): ViTPatchEmbeddings(
 (projection): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
 )
 (dropout): Dropout(p=0.0, inplace=False)
 )
 (encoder): ViTEncoder(
 (layer): ModuleList(
 (0-11): 12 x ViTLayer(
 (attention): ViTAttention(
 (attention): ViTSelfAttention(
 (query): Linear(in_features=768, out_features=768, bias=True)
 (key): Linear(in_features=768, out_features=768, bias=True)
 (value): Linear(in_features=768, out_features=768, bias=True)
 (dropout): Dropout(p=0.0, inplace=False)
 )
 (output): ViTSelfOutput(
 (dense): Linear(in_features=768, out_features=768, bias=True)
 (dropout): Dropout(p=0.0, inplace=False)
 )
 )
 (intermediate): ViTIntermediate(
 (dense): Linear(in_features=768, out_features=3072, bias=True)
 (intermediate_act_fn): GELUActivation()
 )
 (output): ViTOutput(
 (dense): Linear(in_features=3072, out_features=768, bias=True)
 (dropout): Dropout(p=0.0, inplace=False)
 )
 (layernorm_before): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
 (layernorm_after): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
 )
 )
 )
 (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
 (pooler): ViTPooler(
 (dense): Linear(in_features=768, out_features=768, bias=True)
 (activation): Tanh()
 )
 )
 (embed_video_fc): Linear(in_features=768, out_features=512, bias=True)
 (position_encoder_video): Embedding(60, 512)
 (phi): PhiModel(
 (embed_tokens): Embedding(51200, 2560)
 (embed_dropout): Dropout(p=0.0, inplace=False)
 (layers): ModuleList(
 (0-31): 32 x PhiDecoderLayer(
 (self_attn): PhiSdpaAttention(
 (q_proj): Linear(in_features=2560, out_features=2560, bias=True)
 (k_proj): Linear(in_features=2560, out_features=2560, bias=True)
 (v_proj): Linear(in_features=2560, out_features=2560, bias=True)
 (dense): Linear(in_features=2560, out_features=2560, bias=True)
 (rotary_emb): PhiRotaryEmbedding()
 )
 (mlp): PhiMLP(
 (activation_fn): NewGELUActivation()
 (fc1): Linear(in_features=2560, out_features=10240, bias=True)
 (fc2): Linear(in_features=10240, out_features=2560, bias=True)
 )
 (input_layernorm): LayerNorm((2560,), eps=1e-05, elementwise_affine=True)
 (resid_dropout): Dropout(p=0.1, inplace=False)
 )
 )
 (final_layernorm): LayerNorm((2560,), eps=1e-05, elementwise_affine=True)
 )
 (embed_query_fc): Linear(in_features=2560, out_features=512, bias=True)
 (transformer_encoder): TransformerEncoder(
 (layers): ModuleList(
 (0-4): 5 x TransformerEncoderLayer(
 (self_attn): MultiheadAttention(
 (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
out_features=512, bias=True)
 )
 (linear1): Linear(in_features=512, out_features=2048, bias=True)
 (dropout): Dropout(p=0.1, inplace=False)
 (linear2): Linear(in_features=2048, out_features=512, bias=True)
 (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
 (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
 (dropout1): Dropout(p=0.1, inplace=False)
 (dropout2): Dropout(p=0.1, inplace=False)
 )
 )
 )
 (transformer_decoder): TransformerDecoder(
 (layers): ModuleList(
 (0-4): 5 x TransformerDecoderLayer(
 (self_attn): MultiheadAttention(
 (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
out_features=512, bias=True)
 )
 (multihead_attn): MultiheadAttention(
 (out_proj): NonDynamicallyQuantizableLinear(in_features=512,
out_features=512, bias=True)
 )
 (linear1): Linear(in_features=512, out_features=2048, bias=True)
 (dropout): Dropout(p=0.1, inplace=False)
 (linear2): Linear(in_features=2048, out_features=512, bias=True)
 (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
 (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
 (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
 (dropout1): Dropout(p=0.1, inplace=False)
 (dropout2): Dropout(p=0.1, inplace=False)
 (dropout3): Dropout(p=0.1, inplace=False)
 )
 )
 )
 (span_predictor): Linear(in_features=512, out_features=4, bias=True)
)
LORA target modules: ['transformer_encoder.layers.0.self_attn.out_proj',
'transformer_encoder.layers.0.linear1', 'transformer_encoder.layers.0.linear2',
'transformer_encoder.layers.1.self_attn.out_proj',
'transformer_encoder.layers.1.linear1', 'transformer_encoder.layers.1.linear2',
'transformer_encoder.layers.2.self_attn.out_proj',
'transformer_encoder.layers.2.linear1', 'transformer_encoder.layers.2.linear2',
'transformer_encoder.layers.3.self_attn.out_proj',
'transformer_encoder.layers.3.linear1', 'transformer_encoder.layers.3.linear2',
'transformer_encoder.layers.4.self_attn.out_proj',
'transformer_encoder.layers.4.linear1', 'transformer_encoder.layers.4.linear2',
'transformer_decoder.layers.0.self_attn.out_proj',
'transformer_decoder.layers.0.multihead_attn.out_proj',
'transformer_decoder.layers.0.linear1', 'transformer_decoder.layers.0.linear2',
'transformer_decoder.layers.1.self_attn.out_proj',
'transformer_decoder.layers.1.multihead_attn.out_proj',
'transformer_decoder.layers.1.linear1', 'transformer_decoder.layers.1.linear2',
'transformer_decoder.layers.2.self_attn.out_proj',
'transformer_decoder.layers.2.multihead_attn.out_proj',
'transformer_decoder.layers.2.linear1', 'transformer_decoder.layers.2.linear2',
'transformer_decoder.layers.3.self_attn.out_proj',
'transformer_decoder.layers.3.multihead_attn.out_proj',
'transformer_decoder.layers.3.linear1', 'transformer_decoder.layers.3.linear2',
'transformer_decoder.layers.4.self_attn.out_proj',
'transformer_decoder.layers.4.multihead_attn.out_proj',
'transformer_decoder.layers.4.linear1', 'transformer_decoder.layers.4.linear2']
LORA modules to save: ['embed_video_fc', 'embed_query_fc',
'position_encoder_video', 'span_predictor']
trainable params: 2,270,212 || all params: 2,775,739,912 || trainable%:
0.08178763399933416
```

8. Contributions and Acknowledgements

This paper was equally produced by all three authors Prathik, Ranajit, and Jay. Ranajit and Jay were primarily responsible for literature review and initial design of the proposed architecture. Prathik synthesized the findings from the literature review and leveraged his knowledge of PyTorch to prototype and train the final NLVL-DETR model. All three authors then collaborated together in the writing of this final report.

9. References

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer, 2021.

Ivana Balažević, Yuge Shi, Pinelopi Papalampidi, Rahma Chaabouni, Skanda Koppula, and Olivier J. Hénaff. Memory consolidation enables long-context video understanding. 2024.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. June 2015.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision. *CoRR*, abs/2006.13256, 2020. URL `https://arxiv.org/abs/2006.13256`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL `https://arxiv.org/abs/2010.11929`.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. 2017.

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021.

Haroon Idrees, Amir R. Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, February 2017. ISSN 1077-3142. doi: 10.1016/j.cviu.2016.10.018. URL `http://dx.doi.org/10.1016/j.cviu.2016.10.018`.

Cristian Rodriguez-Opazo, Edison Marrese-Taylor, Fatemeh Sadat Saleh, Hongdong Li, and Stephen Gould. Proposal-free temporal moment localization of a natural-language query in video using guided attention, 2020.

Cristian Rodriguez-Opazo, Edison Marrese-Taylor, Basura Fernando, Hiroya Takamura, and Qi Wu. Memory-efficient temporal moment localization in long videos. pages 1909–1924, May 2023. doi: 10.18653/v1/2023.eacl-main.140. URL `https://aclanthology.org/2023.eacl-main.140`.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. *CoRR*, abs/1604.01753, 2016. URL `http://arxiv.org/abs/1604.01753`.

Jing Wang, Aixin Sun, Hao Zhang, and Xiaoli Li. Msdetr: Natural language video localization with moment-moment interaction. 2023.

Shaoning Xiao, Long Chen, Songyang Zhang, Wei Ji, Jian Shao, Lu Ye, and Jun Xiao. Boundary proposal network for two-stage natural language video localization. *CoRR*, abs/2103.08109, 2021. URL `https://arxiv.org/abs/2103.08109`.

Chenlin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers, 2022.

Jingran Zhang, Fumin Shen, Xing Xu, and Heng Tao Shen. Temporal reasoning graph for activity recognition, 2019.

Minghang Zheng, Yanjie Huang, Qingchao Chen, and Yang Liu. Weakly supervised video moment localization with contrastive negative sample mining. 36(3):3517–3525, 2022.

Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression, 2019.