University of Hamburg

LT Research Group

Prof. Chris Biemann and Florian schneider

Video-LLaMA for Text-Video Retrieval

Author:
Narges Baba Ahmadi

October 2023

Contents

1 Introduction

My summer internship with the Language Technology research group was dedicated to the application of state-of-the-art multi-modal Large Language Models for text-video retrieval, with a primary focus on the MSR VTT dataset [1]. Central to our research was the utilization of Video-LLaMA [2], an intricate framework that integrates a Vision-Language Branch and the Audio-Language Branch, endowing LLMs with the capacity to comprehend both visual and auditory elements within video content. It's essential to clarify that, for our internship project, we made use of the existing Video-LLaMA framework, which was forked from the original repository. Our contribution involved the incorporation of this framework into our research, with the aim of enhancing text-video retrieval capabilities.
The internship unfolded in two distinct phases. Firstly, we developed a specialized evaluation script tailored for assessing text-video retrieval systems and analyzed the performance of the X-CLIP [3] as a video retrieval system. In the subsequent phase, we harnessed the capabilities of Video-LLaMA framework to explore its potential in the realm of video retrieval.
In particular, we leveraged different components of Video-LLaMA to calculate embeddings of videos and texts. These embeddings are thought to be served as the foundation for creating a robust video-retrieval system. This approach allowed us to harness the power of multi-modal understanding, enabling our system to comprehend the content of videos and texts, thus facilitating efficient and accurate text-video retrieval.
Furthermore, after obtaining these embeddings through the Video-LLaMA model, we delved into the domain of constrastive multi-modal representation learning. We experimented with training networks that take these embeddings as input and attempt to align the text embeddings and the video embeddings in the same vector space so that we can use them to compute similarities for information retrieval. Our research involved exploring various network architecturee, including transformers [4], and tuning different parameters to optimize the alignment process.
This report serves as a comprehensive chronicle of our endeavors aimed at advancing the field of cross-modal text-video information retrieval. It showcases our in-depth exploration of Video-LLaMA's components and their integration into a practical application, highlighting the potential for advancements in the field of multi-modal language understanding and text-video retrieval.

2 Evaluation Phase

During the evaluation phase of this research project, our primary objective was to evaluate different variants of X-CLIP model, with a particular focus on the MSR-VTT dataset. The ultimate goal was the development of a comprehensive evaluation metric capable of computing vital retrieval performance metrics, including Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and hit rate. To facilitate the execution of this phase, a dedicated virtual environment was meticulously constructed, and it is imperative to activate this environment before running the code.
All essential code and dependencies for this phase have been thoughtfully made available on our GitHub repository, including the forked Video-LLaMA framework. It is pertinent to note that the MSR-VTT dataset, housing both video and text embeddings generated by the specific text-video retrieval model under evaluation, constituted the foundational dataset for this phase of the project.
The evaluation metric, responsible for the computation of retrieval performance metrics, is encapsulated within the 'eval' branch of our GitHub repository, which comprises five distinct files, each assigned a specific role:

- **df-maker.py:** This script takes inputs such as 'video-dir,' 'df-dir,' 'emb-vid-dir,' 'emb-txt-dir,' and 'save-path-dir.' Notably, 'df-dir' denotes the dataframe housing captions for each video, while 'emb-vid-dir' and 'emb-txt-dir' refer to the directories housing video and text embeddings, respectively. The script's primary function is to generate the requisite dataframe necessary for the evaluation metric.
- **main-code.py:** Serving as the central component, this script is executed following the preparation of the final dataframe. It steers the evaluation metric and yields comprehensive performance metrics for the selected text-video retrieval model.
- **video-embedding-cal.py:** This script is specifically designed for recalculating video embeddings as needed.
- **text-embedding-cal.py:** In a similar vein, this script is employed to recalculate text embeddings if deemed necessary.

In the course of the evaluation, this metric was utilized to assess various variations of the X-CLIP text-video retrieval model on the MSR-VTT dataset. The specific results arising from this evaluation, which offer valuable insights into the model's performance, are meticulously documented below.

2.1 Xclip Performance

Retrieval Hit Rate(T2V)

Model	K	Score
xclip-base-patch32	10	0.2580
xclip-base-patch32	5	0.1930
xclip-base-patch32	2	0.1330
xclip-base-patch32	1	0.0860
xclip-base-patch16	10	0.3420
xclip-base-patch16	5	0.2750
xclip-base-patch16	2	0.1870
xclip-base-patch16	1	0.1280
xclip-large-patch14	10	0.4890
xclip-large-patch14	5	0.3940
xclip-large-patch14	2	0.2730
xclip-large-patch14	1	0.2060

Retrieval Hit Rate(V2T)

Model	K	Score
xclip-base-patch32	10	0.4770
xclip-base-patch32	5	0.3660
xclip-base-patch32	2	0.2550
xclip-base-patch32	1	0.1770
xclip-base-patch16	10	0.5110
xclip-base-patch16	5	0.4060
xclip-base-patch16	2	0.2780
xclip-base-patch16	1	0.1920
xclip-large-patch14	10	0.5120
xclip-large-patch14	5	0.4220
xclip-large-patch14	2	0.2930
xclip-large-patch14	1	0.2150

MRR(T2V)

Model	MRR
xclip-base-patch32	0.1465
xclip-base-patch16	0.2032
xclip-large-patch14	0.2976

MRR(V2T)

Model	MRR
xclip-base-patch32	0.2746
xclip-base-patch16	0.2965
xclip-large-patch14	0.3158

MAP(T2V)

Model	MAP
xclip-base-patch32	0.1465
xclip-base-patch16	0.2032
xclip-large-patch14	0.2976

MAP(V2T)

Model	MAP
xclip-base-patch32	0.2741
xclip-base-patch16	0.2967
xclip-large-patch14	0.3157

3 Exploring the Video-LLaMA Model

Following the successful implementation and evaluation of the text-video retrieval metric in the initial phase of our research, we proceeded to explore the capabilities of the Video-LLaMA model. To facilitate this exploration, the Video-LLaMA repository [1] was forked into our project profile. This strategic move allowed us to leverage and extend the extensive codebase and functionalities offered by this innovative framework.
The repository harbored a wealth of pre-existing code that we adeptly incorporated into our research endeavor. In tandem with these existing resources, we supplemented our efforts with custom coding to extract both video and text embeddings from the Video-LLaMA model.
Utilizing pre-existing code for embedding extraction upon forking the Video-LLaMA repository, we embarked on the task of extracting video and text embeddings, a process essential for our research objectives.

[1]https://github.com/DAMO-NLP-SG/Video-LLaMA

The following scripts and methods were instrumental in this endeavor:

- **Video-embedding-cal.py:**
  - Inputs: video-dir-path, save-dir-path, and a Boolean value for selecting a pooling layer (options included First-element, Max-pooling, Average-pooling, Sum).
  - Functionality: This script harnessed the video embedder before the LLaMA framework, in conjunction with the specified pooling layer, to calculate video embeddings. The resulting embeddings were saved as .npy arrays, with each file named after the corresponding video, and subsequently stored in the designated save-dir-path. Optionally, the script could be configured to save the embeddings before pooling, offering versatility in our research approach.
- **Video-embedding-cal-after-llama.py:**
  - Inputs: video-dir-path and save-dir-path.
  - Functionality: This script was tailored to extract video embeddings after the LLaMA model processing. By providing the video directory path and the destination save directory, we obtained embeddings reflecting the unique characteristics introduced by the LLaMA model.
- **text-embedding-cal.ipynb:**
  - Inputs: df-dir, save-dir-path, and a Boolean value for selecting a pooling layer (options included First-element, Max-pooling, Average-pooling, Sum).
  - Functionality: This Jupyter Notebook script played a pivotal role in extracting text embeddings. It assumed the existence of a dataframe (df) containing video captions, with multiple captions available per video (up to a maximum of 20). The script saved the resulting embeddings in .npy format, mirroring the video names, in the designated save-dir-path. Notably, it is important to emphasize that for videos with multiple captions, the script saved these captions as separate files, using a numbering scheme. For example, for a video with the identifier 7890, the resulting files would be named as follows:
    - video7890-1.npy
    - video7890-2.npy
    - video7890-3.npy

    Similar to the video embedding process, the script allowed for the optional preservation of embeddings before the application of pooling layers.

In summary, the second phase of our research involved extracting video and text embeddings from the Video-LLaMA model by utilizing a combination of pre-existing code and custom scripting. This phase was critical in our exploration of the model's capabilities and laid the foundation for subsequent analyses and experiments.

3.1 Evaluation of Video-LLaMA Embeddings on MSR-VTT Dataset Using MRR:

Following the successful extraction of text and video embeddings utilizing the Video-LLaMA model, the next phase of our research focused on the evaluation of these embeddings within the context of the MSR-VTT dataset. In this evaluation, we aimed to assess the quality of the embeddings with a particular emphasis on their suitability for text-video retrieval tasks. To achieve this, we adopted the Mean Reciprocal Rank (MRR) as our primary evaluation metric. MRR is a robust measure for evaluating the effectiveness of retrieval systems, and we chose it as the benchmark for assessing the performance of our embeddings. It provides valuable insights into the retrieval capabilities of our approach.
The resulting MRR scores for Video-LLaMA(T2V), obtained from this evaluation, were stored in a result dataframe. This dataframe is accessible at the following path:

```
/home/ahmadi/video-ir/dataset/llama_data/after_pooling/test_data/results_dataframe.csv
```

The results were as below:

Video Pooling Layer	**Text Pooling Layer**	**MRR**
averagePooling	averagePooling	0.0245
averagePooling	firstElement	0.0075
averagePooling	maxPooling	0.0103
averagePooling	sum	0.0245
firstElement	averagePooling	0.0072
firstElement	firstElement	0.0075
firstElement	maxPooling	0.0066
firstElement	sum	0.0072
maxPooling	averagePooling	0.0109
maxPooling	firstElement	0.0075
maxPooling	maxPooling	0.0095
maxPooling	sum	0.0109
sum	averagePooling	0.0245
sum	firstElement	0.0075
sum	maxPooling	0.0103
sum	sum	0.0245

Evidently, the scores are much worse compared to X-CLIP; however, this was expected since the embeddings are not aligned yet. The goal of the next phase is to align the embeddings. These MRR scores serve as a crucial yardstick for measuring the effectiveness of Video-LLaMA embeddings and their potential to enhance text-video retrieval capabilities.

3.2 Aligning the Embeddings Using Contrastive-Learning Methods

In the subsequent phase of our research, our primary objective was to determine whether providing the embeddings to neural networks would lead to a reduced distance between the text and video embeddings. The models we experimented with are as below:

3.2.1 Custom MLP Model

Below you can find the components of our MLP models:

- **Input Layer:** Serving as the inception point, this layer received the initial text and video embeddings.The shape of it was initially set to 4096 due to the shape of our embeddings.
- **Hidden Layers:** These encompassed multiple layers, featuring linear transformations, Rectified Linear Unit (ReLU) activations, Layer Normalization, and dropout layers. These elements collaboratively enabled the model to learn and refine the embeddings.
- **Output Layer:** This concluding layer generated the definitive embeddings post-processing.
- **Cosine Similarity:** To evaluate the similarity between the transformed text and video embeddings, we relied on the cosine similarity metric.
- **Optimizer:** The model's parameters were fine-tuned through the use of the AdamW optimizer.

The details of different MLP architectures used in this phase can be found in section 3.3 .

3.2.2 Custom Transformer Model:

Our exploration extended to the Transformer[4] encoder only model, acknowledged for its prowess in managing sequential data, with the architecture tailored to optimize embeddings.

- **Input Layer:** Analogous to the MLP model, this layer received the initial text and video embeddings.
- **Transformer Encoder:** A stack of TransformerEncoderLayers handled the processing of input embeddings, with users retaining flexibility to specify the number of encoder layers and the number of attention heads (nhead).
- **Output Layer:** The model's output layer delivered the refined embeddings following the transformation.
- **Cosine Similarity:** As in the MLP model, cosine similarity was employed to assess the similarity between the transformed text and video embeddings.
- **Optimizer:** Consistency was maintained as we continued to utilize the AdamW optimizer for the systematic adjustment of model parameters.

3.2.3 Other Supporting Scripts:

In conjunction with our neural network models, several additional scripts played pivotal roles in our research, all orchestrated within the realm of **PyTorch Lightning** [2]:

- **create-df.py:** This script served as the linchpin in generating dataframes from text and video embeddings directories. It followed a series of essential steps, beginning with the 'generate-and-save-dataframes' function to create the dataframe. Subsequently, the 'double-and-shuffle-dataframes' function augmented the dataframe, ensuring the inclusion of both negative and positive examples. Lastly, the 'process-and-save-dataframe' function facilitated the division of training and validation examples. Importantly, this script demanded that separate directories be maintained for different pooling layers within text and video embedding directories; failure to do so would trigger an error. Additionally, a specific naming convention was enforced in the 'generate-and-save-dataframes' function, utilizing information from the directories to structure the output CSV path.
- **callbacks.py:** This script incorporated custom callbacks for printouts before and after training, offering insights to monitor and evaluate the training process.
- **checkloader.py:** Used to verify the proper operation of our data loader, this script ensured the seamless and reliable flow of data for our experiments.
- **config.py:** A central repository for configurations and settings integral to our research, ensuring uniformity and reproducibility across experiments.
- **dataset.py:** This module was instrumental in managing data points during the training phase. It efficiently handled the loading and processing of data, guaranteeing compatibility with the neural network models.
- **lightning-dm.py:** Housing the MSR-VTT-Datamodule class, this script acted as the bridge between our dataset and the PyTorch Lightning framework, simplifying data management.
- **model.py:** The architectural blueprint of our MLP model was documented in this file, providing the foundation for our experiments.
- **transformer.py:** This script delineated the architecture of our custom Transformer model.

3.3 Experimental Findings

Our experimentation included an assessment of both the custom MLP and Transformer models, with a particular focus on aligning the embeddings. Notably, we observed that the choice of pooling layers did not yield substantial variations in MRR scores but choosing Average pooling layer for both text and video embeddings worked better so we chose it for all experiments.

[2] https://lightning.ai

In the upcoming sections of this report, we will delve deeper into the results and insights extracted from our experiments. These findings represent significant strides in our quest to optimize embeddings through neural network-based enhancements.

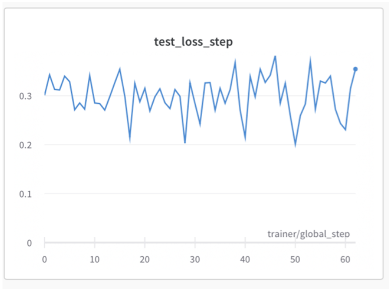
3.3.1 Experiments with MLP

In this section, we employ the Multilayer Perceptron (MLP) as the chosen model for our experiments. The loss function adopted for these experiments is the Mean Squared Error (MSE) Loss.

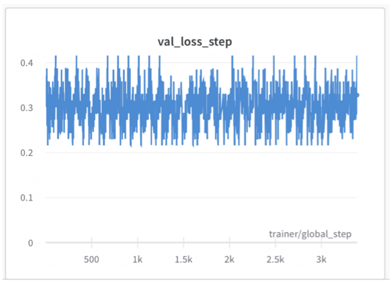
Base experiment:

The initial experiment can be characterized as the most elementary within this study. For this experiment, we employed a single positive and a single negative caption per video and trained a straightforward neural network. The model was trained under the following specified parameters:

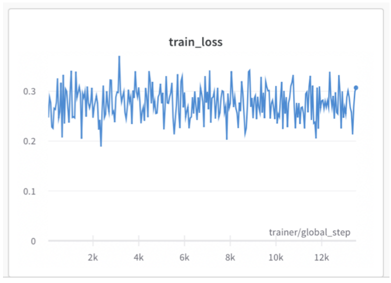
- layers: [4096, 4096]
- learning rate = 0.001
- batch size = 32
- number of epochs = 30
- precision = 64



(a) Test Loss



(b) Validation Loss



(c) Train Loss

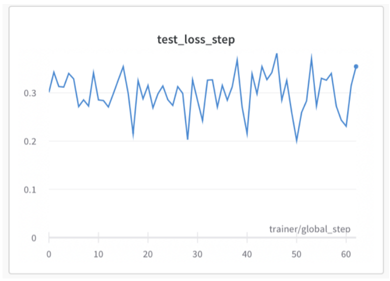
Figure 1: Loss plots

Augmenting Training Data:

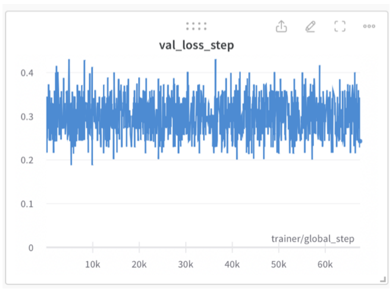
Evidently, the preceding experiment yielded inconclusive results. To address this, we opted to augment our dataset in the subsequent experiment by incorporating a more substantial quantity of training examples. Specifically, we employed 20 positive and 20 negative captions per video, thus increasing the diversity and volume of the training dataset. The model was trained under the following prescribed parameters:

- layers: [4096, 4096]
- learning rate = 0.001
- batch size = 32

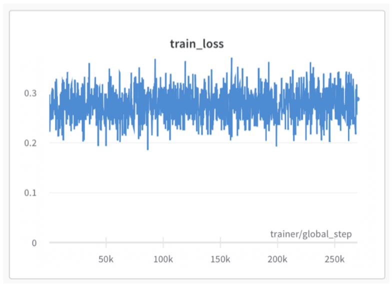
- number of epochs = 30
- precision = 64



(a) Test Loss



(b) Validation Loss

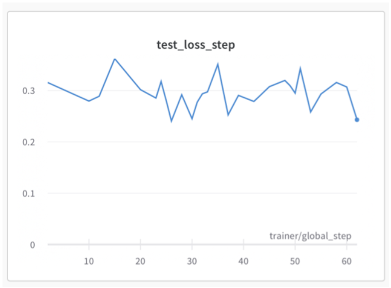


(c) Train Loss

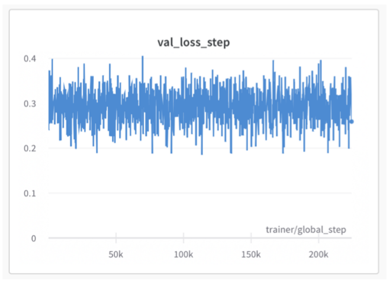
Figure 2: Loss plots

Employing a More Complex Network Architecture:
As using more training examples did not work either, we tried changing the architecture of our network. 20 positive and 20 negative captions per video were used. We also increased the number of epochs. we trained the model with the following parameters:

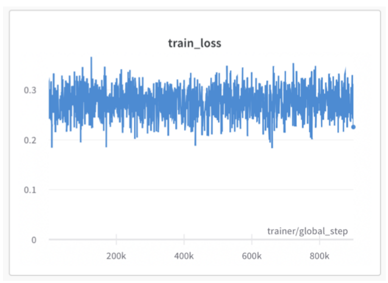
- layers: [4096, 2048, 1024, 1024, 512, 512]
- learning rate = 0.001
- batch size = 32
- number of epochs = 100
- precision = 32



(a) Test Loss



(b) Validation Loss



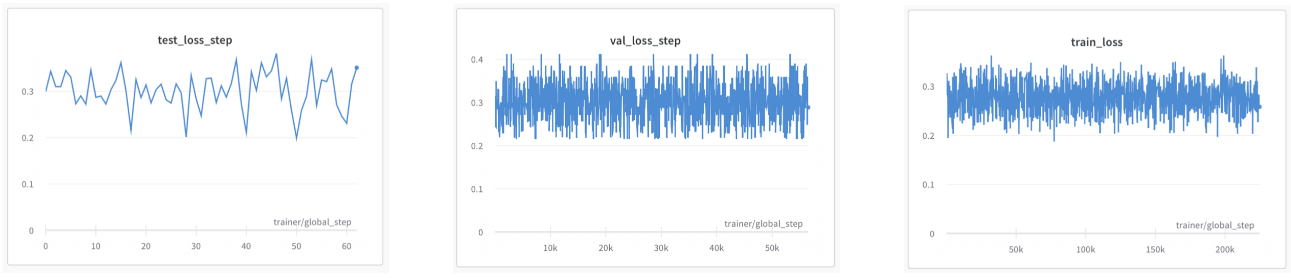
(c) Train Loss

Figure 3: Loss plots

Extending Training Duration: Maximizing the Number of Epochs
In this experimental iteration, our objective was to explore the effects of prolonged training on the neural network's learning capacity. To this end, we conducted an experiment with a significantly increased number of training epochs, specifically 500 epochs. The primary aim was to assess whether the network would exhibit any learning trends or improvements in later epochs. The model was trained under the following specified parameters:

- layers: [4096, 2048, 1024, 1024, 512, 512]
- learning rate = 0.001
- batch size = 32
- number of epochs = 500
- precision = 32

The results were as below:



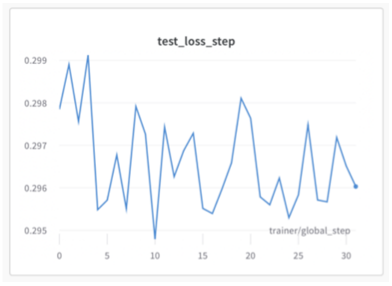
(a) Test Loss (b) Validation Loss (c) Train Loss

Figure 4: Loss plots

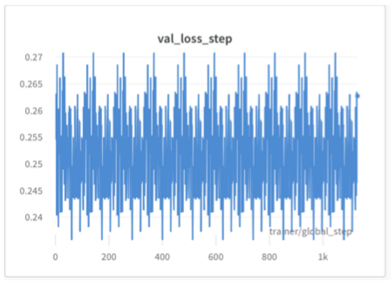
3.3.2 Experminets with Transformer

Base experiment:
In the pursuit of more effective model architectures, we explored the application of Transformer models as an alternative to the Multilayer Perceptron (MLP) approach. The decision to transition to Transformers was prompted by the observation that, even after 500 epochs of training, the MLP appeared to lack the capacity to sufficiently learn from our data, suggesting the need for a more complex model.
In this phase of our experimentation, we recalculated the embeddings and refrained from employing pooling layers, resulting in video embeddings of shape (32,4096) and text embeddings with varying shapes contingent on caption length, i.e., (5,4096), (9,4096), and so forth. To address the heterogeneity in embeddings, we leveraged the use of the [CLS] tokens. This particular experiment was conducted using a dataset that included one positive and one negative example per video. The model was trained under the following specified parameters:

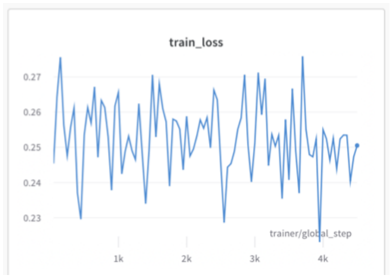
- num encoder layers = 6
- nhead = 8
- learning rate = 0.001
- batch size = 32
- number of epochs = 10
- precision = 64



(a) Test Loss



(b) Validation Loss

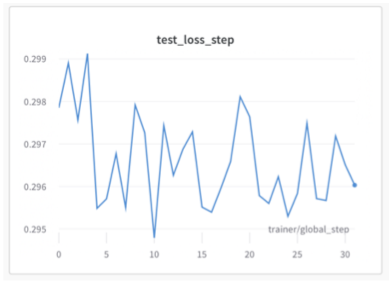


(c) Train Loss

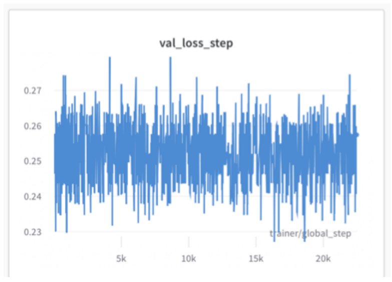
Figure 5: Loss plots

Augmenting the Training Dataset:
It is apparent that the Transformer model did not demonstrate substantial learning progress, even after the transition from the MLP architecture. This observation prompted us to investigate the influence of dataset augmentation by increasing the number of examples. In this experimental phase, we incorporated 20 positive and 20 negative examples for each video, with the aim of examining whether this augmentation would lead to improved learning. The model was trained under the following specified parameters:

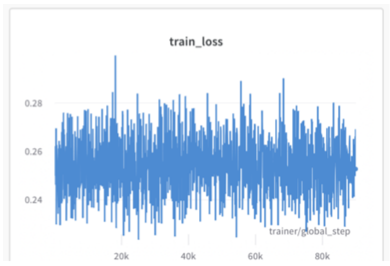
- num encoder layers = 6
- nhead = 8
- learning rate = 0.001
- batch size = 32
- number of epochs = 10
- precision = 64



(a) Test Loss



(b) Validation Loss



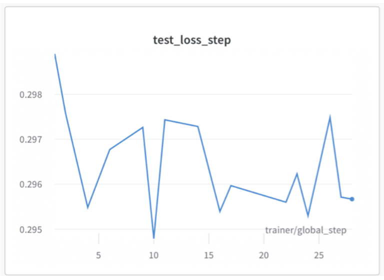
(c) Train Loss

Figure 6: Loss plots

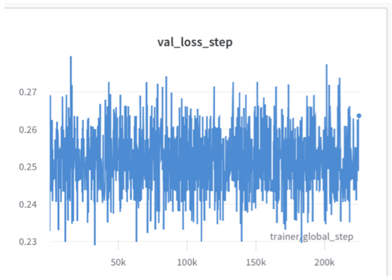
Extending Training Duration: Maximizing the Number of Epochs:
In this experiment, our focus shifted towards exploring the effects of extended training duration on the Transformer model's ability to discern underlying patterns. We aimed to ascertain whether the model, in the course of training, would unveil latent patterns and

demonstrate improved learning in later epochs. The model was trained under the following specified parameters:

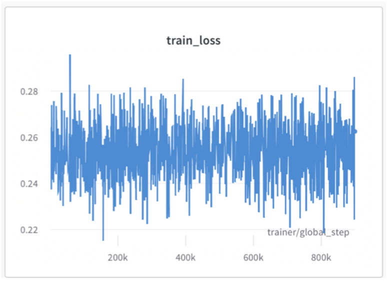
- num encoder layers = 6
- nhead = 8
- learning rate = 0.001
- batch size = 32
- number of epochs = 100
- precision = 64



(a) Test Loss



(b) Validation Loss

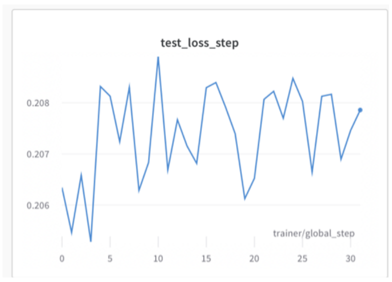


(c) Train Loss

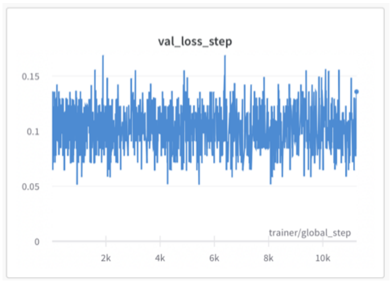
Figure 7: Loss plots

Exploring Alternative Loss Function:
In light of the model's limited learning progress after 100 epochs, we conjectured that the choice of the loss function might not have been suitable for our objectives. Consequently, we embarked on an exploration of an alternative loss function. Specifically, we transitioned to using the contrastive loss function and conducted a subsequent round of training.
For this experiment, our dataset consisted of 20 negative and 20 positive examples for each video. The parameters for this experiment were as follows:

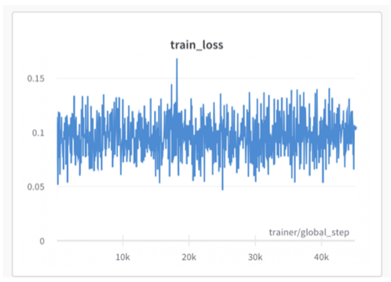
- num encoder layers = 6
- nhead = 8
- learning rate = 0.001
- batch size = 32
- number of epochs = 5
- precision = 64



(a) Test Loss



(b) Validation Loss

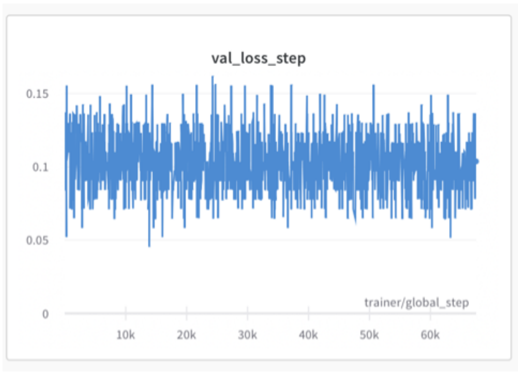


(c) Train Loss

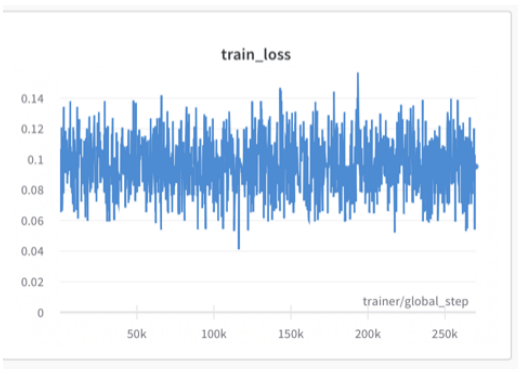
Figure 8: Loss plots

Adjusting Learning Rate for Model Training:
In response to the observed challenges in the training process, where the model exhibited limited learning progress, we conjectured that the learning rate might have been a contributing factor. To address this concern, we implemented a reduction in the learning rate and conducted a subsequent training iteration.
The key parameters employed for this experiment were as follows:

- num encoder layers = 6
- nhead = 8
- learning rate = 0.00001
- batch size = 32
- number of epochs = 5
- precision = 64



(a) Validation Loss

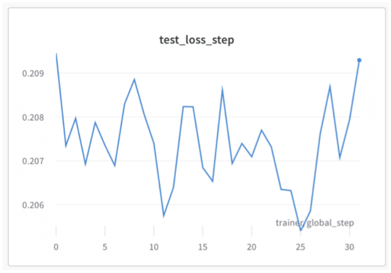


(b) Train Loss

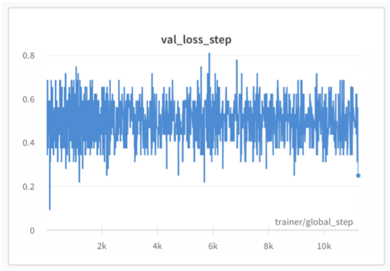
Figure 9: Loss plots

Reconsidering Video Embeddings:
After doing all those experiments, we though maybe we have to use the embeddings after llama model rather than before it. So, as our last experiment, we calculated the embeddings again and trained the network.

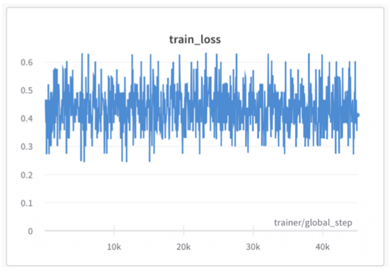
- num encoder layers = 6
- nhead = 8
- learning rate = 0.00001
- batch size = 32
- number of epochs = 5
- precision = 64



(a) Test Loss



(b) Validation Loss



(c) Train Loss

Figure 10: Loss plots

4 Conclusion:

In the realm of multi modal research, this project embarked on a journey to harness the immense potential of the Video-LLaMA model for advancing the field of text-video retrieval. With the initial success of obtaining video and text embeddings from this innovative model, our subsequent endeavors aimed to enhance these embeddings to build an effective text-video retrieval system.
Our exploration encompassed not only the video embeddings but also text embeddings, with a goal to bridge the gap between textual descriptions and visual content. While we successfully extracted these embeddings, we encountered challenges in aligning them to achieve a high-performing retrieval system.
Despite numerous experiments involving diverse neural network models, varying loss functions, and adjustments to training parameters, our journey did not yield the breakthrough we initially sought. The complexities of the task became apparent, as the optimization of embeddings for text-video retrieval proved to be a multifaceted challenge.
In our quest to understand the factors contributing to the inconclusive results, we hypothesize that the Video-LLaMA model's different layers may capture distinct linguistic properties. These layers may be responsible for the extraction of various features, ranging from surface form attributes to coreference patterns. To address this, we must reconsider our approach to extracting embeddings, potentially from different layers of the model, which opens up an exciting avenue for future exploration.
Besides, it is crucial to note the unique challenges posed by video-llama and similar language models, which predominantly operate as decoder-only models. Unlike encoder-only models

that compute full self-attention, these models employ masked backward-attention. This presents a significant challenge since encoder-only models are conventionally used for generating high-quality embeddings. To the best of our knowledge, our attempt to compute embeddings from decoder-only models, such as Video-LLaMA, represents a pioneering effort in this domain.
The investigation into alternative loss functions and dimensions for embeddings projection has provided valuable insights, despite not leading to immediate success. It is evident that we need to explore more avenues, each offering a unique perspective and potential solution.
In summary, this project represents a challenging but rewarding journey into the realm of text-video retrieval, where the aim is to bridge the linguistic and visual domains. While our experiments did not yield the desired breakthrough, they illuminated the complexity of the task and the need for continued exploration. The quest to enhance text-video retrieval remains a priority, driven by the understanding that innovative solutions often emerge from persistent inquiry and creative problem-solving. The future of multimedia retrieval is rich with opportunities for innovation, and our commitment to advancing this field remains unwavering.

References

[1] J. Xu, T. Mei, T. Yao and Y. Rui, "MSR-VTT: A Large Video Description Dataset for Bridging Video and Language," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 5288-5296, doi: 10.1109/CVPR.2016.571.

[2] H. Zhang, X. Li, and L. Bing, "Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding," arXiv preprint arXiv:2306.02858, 2023.

[3] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling, "Expanding Language-Image Pretrained Models for General Video Recognition," arXiv preprint arXiv:2208.02816, 2022.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," arXiv:1706.03762 [cs.CL], 2017. https://doi.org/10.48550/arXiv.1706.03762