Ego-centric Videos and Natural Language Queries

Leone Aurora
s334258@studenti.polito.it

Narese Michele
s329892@studenti.polito.it

Racca Riccardo
s315163@studenti.polito.it

Abstract

This report aims to tackle the task of action recognition and identification in egocentric videos, that is, identifying a timeframe where the answer to a natural language query can be seen within the provided video, and then exploiting this knowledge in order to build a textual answer.
Different architectures are implemented and trained on various pre-extracted features of Ego4D dataset [6], whilst multiple metrics are used to compare the performance of different models with the benchmark, in order to obtain a robust model that is employable in the next step.
Subsequently, a video question-answering pipeline is built by leveraging the results provided by the previous architectures and using them as input for a VLM (Video Language Model), to not only retrieve a time interval but also provide a textual answer. Finally, an ensemble method is introduced to obtain better predictions while exploiting the, partially wasted, computational effort needed to build and compare different models.

You can find the code and data used at
https://github.com/Auroraleone/MLDLproject-

1. Introduction

Videos taken from a human point of view have lately become more and more popular as they are employed in many different fields such as wearable computing, augmented reality and human-computer interaction. As a consequence, the urge of developing models, that are able to tackle this category of problems, led to the need of collecting large-scale data in order to better capture the nature of this task. For this reason, datasets such as EpicKitchens [2] and Ego4D [6] were created. Many tasks are related to egocentric video understanding such as action anticipation, cross-modal retrieval or action recognition (both fully and weakly supervised). These challenges involve identifying a triplet of verb, noun, action by observing a video segment, predicting it from a video of unspecified length or generating video-to-text retrieval with high *semantic relevance*.
This paper focuses on the challenge of Ego4D NLQ benchmark, which is defined as a temporal segment prediction task where, given a video clip and a text query in natural language, the model is asked to predict the temporal segment where the answer is visible or deducible. The queries can be related to objects, places, people, and performed activities. The queries are created from a restricted number of possible templates, such as "*In what location did I see object X?*" or "*What X did I Y?*" where X and Y could be verbs or nouns.
As we can see in Figure 1, there's a relevant imbalance in the distribution of NLQ templates.

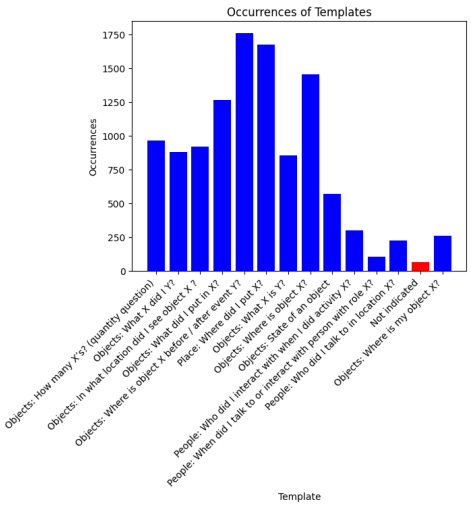


Figure 1. Template distribution in Ego4D data

2. Related Works

2.1. Egocentric Video Datasets

A considerable amount of large-scale video datasets taken from an egocentric point of view has been developed in the last few years, such as EPIC-Kitchens [2], Charades-Ego [4] and, more importantly, Ego4D [6]. The first one focuses solely on kitchen activities, while the others range in many different daily life situations.

The main challenge that arises in creating these kinds of datasets is annotating the ground truth and narrating the events. As the size of the dataset increases, manually annotating and labelling all videos becomes more and more time-consuming, risking quality degradation. The process of narration heavily relies on human effort, requiring individuals to watch (usually multiple times) and summarize the video in a few sentences related to specific temporal segments. This allows models to be trained on different Episodic Memory tasks. Episodic Memory refers to past-dependent knowledge, which means being able to produce a result from a query based on past experiences: if we consider the question "*"what did I eat and who did I sit by on my first flight to France?*", this has to be distinguished with a question such as "*What's the capital of France?*" even though they are semantically related.
The query type naturally defines three different tasks

- Natural Language Queries (NLQ)
- Visual Queries (VQ)
- Moment Queries (MQ)

2.2. Natural Language Video Localization

The task of identifying temporal intervals in a video by using Natural Language Queries has been widely explored in the last years. The main challenge lies on how the two modalities (NLQ and Video) interact, in fact, it is important to model the cross-modal interaction between them. At first, models interpreted this task as a ranking problem, relying on a sort of multimodal matching. Many other implementations have been tested, such as formulating Natural Language Video Localization (NLVL) as a sequence decision-making problem and adopting reinforcement learning-based approaches to progressively observe candidate moments conditioned on the language query. It has also been approached as a regression task to directly predict the target video span. More recently, [10] addresses the problem using the concept of a span-based QA approach, by treating the input video as text passage. In particular, VSLBase (that here is implemented starting from VSLNet) adopts standard span-based QA framework and VSLNet explicitly addresses the differences between NLVL and traditional span-based QA tasks by employing a query-guided highlighting (QGH) strategy, which is further discussed in section 3.1.

2.3. Feature extraction on egocentric videos

Feature extraction on egocentric videos has advanced significantly with the advent of deep learning and large-scale datasets. Traditional handcrafted feature methods have been replaced by deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which are better suited to capturing the unique spatial and temporal dynamics of first-person video. More recently, Video-Language Pretraining (VLP) approaches have further enhanced these representations by leveraging large-scale video-text datasets. Techniques like joint-encoders used in [3], which fuse video and text inputs, and dual-encoders as in [7], which independently project video and text into a shared space, have been particularly effective. The joint-encoder approach enables models like Omnivore to learn representations that are transferable across different visual modalities, enhancing their utility in various visual tasks within the egocentric video domain. The dual-encoder approach, as in EgoVLP, uses positive and negative sampling strategies to improve performance on tasks like video retrieval and question answering. This facilitates efficient indexing and retrieval based on similarity measures, thanks to the dual-encoding strategy.

3. Methodology

In this section, firstly the architectures used and the modifications implemented are presented to provide a broader understanding of the project's objectives.

Then, the final pipeline is described along with its motivation and structure.

Finally, a thorough description of the ensemble method employed is provided.

3.1. VSLBase and VSLNet

The architectures that will be used and fine-tuned will be VSLNet and VSLBase [10]. The reason why they were adopted is that VSLNet and VSLBase are specifically designed for the NLQ task. As shown in Figure 2, both models figure two feature extractors: one used for videos and one for text data to obtain word embeddings (this will be discussed in section 3.2). After this step, both visual and text input are transformed along with the answer span into a joint set of triples (**V**, *Q*, **A**) representing (*Context*, *Question*, *Answer*). Afterwards, both embeddings are projected into a common space and encoded using the same feature encoder (both in terms of structure and parameters used). The outputs are then merged using Context-Query Attention to capture the cross-modal interactions between visual and textual features. At this point, VSLNet features a Query-Guided Highlighting (QGH) strategy, that addresses the major differences between text span-based QA and NLVL tasks by considering the target interval as foreground and the rest of the video as background. The foreground boundaries are then extended by a factor α in order to look for possible context indicators that could help focus on subtle differences between video

frames, the QGH could be therefore seen as a binary classifier that has to identify which frames belong to the foreground. In the end, a Conditioned Span Predictor is built by using two unidirectional LSTMs and two feed-forward layer in order to compute the probability distribution of start and end boundaries.

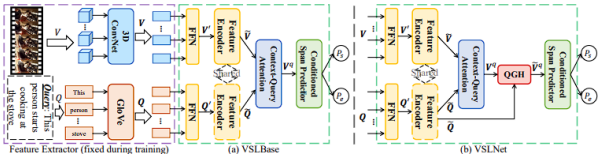


Figure 2. Overview of VSLBase and VSLNet architectures

3.2. GloVe and Bert

In the context of natural language processing (NLP), BERT (Bidirectional Encoder Representations from Transformers) and GloVe (Global Vectors for Word Representation) are distinguished models designed to enhance text comprehension through distinct methodologies.

BERT, developed by Google AI, revolutionized NLP by leveraging transformer models to capture the bidirectional context in text. Its applications span tasks such as sentiment analysis, question answering, and language inference, owing to its ability to grasp intricate language relationships and nuances.

Conversely, GloVe operates as an unsupervised learning algorithm that constructs vector representations for words based on statistical co-occurrence data from extensive text corpora. These embeddings are particularly effective for tasks requiring semantic understanding and analogical reasoning at the word level.

The transition from BERT to GloVe involves utilizing pre-trained BERT models to extract contextualized embeddings of words or sentences. These embeddings, which encapsulate rich syntactic and semantic details, can be adapted or aligned with GloVe embeddings. This process is valuable when exploring different text encoding strategies, leveraging GloVe's static embeddings that derive from statistical properties rather than contextual nuances.

In the context of our project, the objective is to experiment and analyse how the change of the feature extractor, from the typical BERT-based text encoder to GloVe embeddings, for text data in VSLNet and VSLBase architectures will impact the overall performances. This adaptation involves preprocessing textual inputs to convert them into GloVe embeddings using available pre-trained models.

Through this exploration, the aim is to distinguish the subtle differences in results obtained using GloVe embeddings compared to BERT embeddings within the VSLNet architecture. This investigation might provide insights into how varying text encoding strategies influence the model's capability to interpret and respond to natural language queries posed over egocentric video data.

3.3. EgoVLP and Omnivore

Since training models on video datasets is often impractical due to computational requirements, it's imperative to use a "synthetic representation" of the videos present in Ego4d dataset.

For this reason, many video-language models have been pre-trained on the full dataset in order to extract features that serve as compact representations of the video itself.

This project utilizes features pre-extracted by [3] and [7]. The first employs a single model capable of processing various visual modalities, extracting rich visual features by comprehending both spatial and temporal aspects of the data. The second is specifically designed for video-language tasks and pre-trained on large-scale egocentric datasets like Ego4D, this enables it to generate highly effective representations for tasks such as video captioning, video question answering and video-language interactions.

Two distinct representations have been implemented and tested to ensure a more comprehensive analysis, and the results will be thoroughly examined.

3.4. Building a video QA pipeline

The NLQ challenge aims at identifying a temporal interval where the answer of a NLQ is clearly visible. However, this still requires watching the retrieved interval in order to get a textual answer. Many VLM models such as [8] tend to struggle when dealing with long and unstructured videos, especially in terms of computational effort due to the large amount of video tokens that are, for the majority, unrelated with the answer.

Acknowledging these issues necessitated finding a way to overcome them by leveraging both models to create a unified pipeline in order to provide a unique model that, starting from a long and unstructured video and an input query, is able to produce a textual answer (along with the temporal interval associated if needed).

As shown in Figure 3, starting from a clip from the NLQ challenge in Ego4d where the (almost) correct video interval is retrieved by the best model obtained, the original video is taken and trimmed in correspondence with the predicted interval. The output segment is then fed to Video-LLaVa in order to retrieve the answer desired.

3.5. Ensemble method on the NLQ task

An "ensemble" method is proposed, where the results of different models are combined to obtain a more accu-

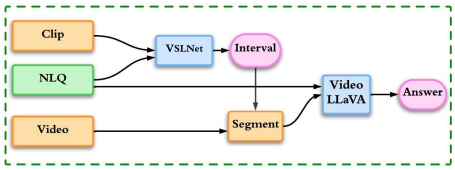


Figure 3. Pipeline for video QA

rate one.

3.5.1 Reasons

In complex ML problems choosing the appropriate model might be challenging. The model training and selection process often involves a grid-search, aiming for the best model by tuning hyperparameters and evaluating the outcomes accordingly. This computational effort becomes useless once the best model is found and all the evaluations are thrown. The following method exploits this necessary effort to obtain potentially better predictions.

3.5.2 Models

All models used are neural networks, each one trained over 10 or more epochs, providing a large number of predictions ready to be combined. Since the quality of predictions is expected to improve as epochs increase, only the ones provided by the last three epochs of each model are considered.

3.5.3 Metrics

The results of the models are structured as matrices, where the row index corresponds to the interval being predicted, whilst five columns contain the predictions proposed by the model for that interval, in decreasing order of confidence.

Three measures of goodness and accuracy of the model are proposed: "R@n, IoU = μ", "mIoU" following [10], [5] and [9] and "mIoU_m". The last is a little variation from "mIou", where the mean of IoUs is computed for each row, considering the first $m \leq 5$ predictions and obtaining a vector whose average is the mIoU_m. This helps to focus and evaluate the model considering just a subset of its prediction, eventually just the first one. For the "R@n, IoU = μ" the results are reported for $n \in \{1, 5\}$, $\mu \in \{0.3, 0.5\}$, while the "mIoU_m" is shown for $m = 1$.
After choosing the preferred metric, a vector of weights is available for the models, as these measures are desirable to be large, the best one will have the highest measure value and so on.
The "influential" models, i.e. those that are sufficiently close to the best, are selected and the convex combination is built.

3.5.4 Convex combination

Two approaches are presented. One simple and intuitive, providing a general explanation of the idea. And a second one, slightly more complicated.

- First heuristic: (Prior elections) Since these are complex tasks, different models, even good ones, provide predictions that differ significantly. Thus, combining results may lead to intervals that lie in between where there is neither the prediction of one nor the other. To overcome this issue, it is necessary to declare a representative, in this case, the best-performing one, who is willing to engage, only if the ideas being considered are "sufficiently" close to its. Models provide their prediction for each interval to be forecasted, including the newly appointed representative. Subsequently, each proposal is compared with the representative's, and only those predictions, having IoU with the representative one larger than a threshold, are considered for the agreement, constituting the advisors. Finally, after decreeing: the influential models, the representative and the advisors, it follows the comparison, in which the proposals of the advisors are combined in a convex manner, with the just computed weights. The representative model is the focal point of this convex combination which ultimately tends to move little from its proposal. The clear advantage of this heuristic is that by properly adjusting the parameters, one cannot do worse than the representative, which is still the best that would have been chosen, providing a kind of inferior bound. Unfortunately, this method might not properly combine models performing equally well, if just one is picked. Moreover, deviating from the representative is not easy and this is limiting.

- Second heuristic: (Posterior elections) This heuristic closely resembles the one just presented. The sole variations are found in the choice of the representative model, which aims to rectify the limitations of the previous heuristic while preserving its benefits. The idea is to take advantage of situations where two or more models are almost equivalent in terms of performance, and choosing only one among them might be limiting. The process is a voting scheme among the influential models, calling a vote for every interval to be predicted. Once all models have made their predictions, a voting procedure is carried out for each predicted interval. Influential models cast their votes for a prediction if it aligns with theirs, using the IoU. The model whose prediction garners the highest number of votes emerges as the winner and is chosen as

representative for the interval to predict. If two or more models tie, the one with the largest weight is selected. From this point onward, everything proceeds as before for the combination. Once the representative model for the interval to be predicted is chosen, it is kept until the next interval, where a new votation will be conducted.

4. Implementation

4.1. Dataset

The training process and the analysis are conducted on the NLQ Episodic Memory dataset which consists of a set of 1,659 clips with an average length of 8.2 minutes. They are already split in training, validation and test set as shown in Table 1.

Split	Train	Val	Test
#Video hours	136	45	46
#Clips	1.0k	0.3k	0.3k
#Queries	11.3k	3.9k	4.0k

Table 1. NLQ dataset statistics across the train/val/test splits

4.2. NLQ challenge

In order to obtain a robust model that is able to perform NLVL task, the standard formulation of VSLNet [1] is employed at first: it consists of the structure presented in Figure 2 but that uses Bert as default.

To better understand the impact of the QGH module, it has been removed following the construction of VSLBase architecture. Additionally, the loss computation method necessitates be altered by removing the "highlight" component from the loss function

$$\mathcal{L}_{VSLNet} = \mathcal{L}_{loc} + \mathcal{L}_{highlight} \quad (1)$$

$$\mathcal{L}_{VSLBase} = \mathcal{L}_{loc} \quad (2)$$

Where the localization loss is expressed as the Cross-Entropy between the labels for the start and end boundaries compared with the probability distribution of them

$$\mathcal{L}_{loc} = \frac{1}{2}[f_{CE}(Y_s, \mathcal{P}_s) + f_{CE}(Y_e, \mathcal{P}_e)] \quad (3)$$

The highlight component is typically used to encourage the model to pay more attention to specific regions within the video that are more relevant to the query. By computing the loss without this component, the model transitions from VSLNet to VSLBase, a simplified version of the model.

The implementation is further modified by switching from BERT to GloVe for the text encoder. This change consists in replacing BERT embeddings with GloVe ones, therefore the way textual features are represented and processed within the model is changed in order to understand the impact that a given text encoder (frozen during the training process) has on the overall performances on the NLQ task.

The third modification is obtained by training the model (VSLNet or VSLBase) with egoVLP pre-extracted features instead of the ones obtained from Omnivore that are already available from Ego4D CLI. This involves changing how the model "perceives" the video by giving as an input a different representation of it.

Every combination is then trained with the baseline parameters reported in Table 2. Once this first training phase is completed, the results and performances obtained are then analysed in order to elect "the most promising architecture" (see section 4.5). The best model so far has been employed in the QA pipeline described in the next paragraph and the impact of the hyperparameters has been further analysed in order to obtain a fine-tuned model.

Parameter	Value
epochs	10
batch size	32
hidden dimension	128
highlight λ (only for VSLNet)	5.0
initial learning rate	0.0025

Table 2. Parameters used for training the baseline models

The metrics used for training are the ones presented in section 3.5.3 as they are the ones mostly used in the literature such as [10] and are specifically designed to deal with overlapping intervals.

4.3. Video QA pipeline

Once the results provided by the models addressing the NLVL task are satisfactory, the best model obtained is used to predict intervals associated with the validation data. The results are then examined, and a subset of positive examples (those with an IoU on the first predicted interval above a certain threshold) is selected for the next step. All information related to each sample is stored in a database.

Each clip associated with the samples selected is part of a video, that is downloaded from Ego4D dataset using the provided tool. Once the videos are available, ffmpeg is used in order to obtain the corresponding segment derived from the model prediction.

The segments, along with their associated queries, are then fed into a Video-Language Model (specifically Video LLaVa). This model produces the textual response associated with the input query by analyzing the provided video frames. Some interesting qualitative results are reported below in Figure 4.

Q: Where was the knife before I picked it?
A: The knife was on the table before I picked it up.

(a) First example

Q: What did I pour in the frying pan?
A: I poured some oil in the frying pan and added some seasoning.

(b) Second example

Figure 4. Answers of the QA pipeline

4.4. Ensemble

In this paragraph, rather than a detailed discussion on the raw implementation of the anticipated ideas, the focus is on listing and explaining the hyper-parameters of the ensemble model and reporting the values used in the grid-search. The set of input parameters is: {*"num_mod"*, *"split"*, *"rank"*, *"inter"*, *"miou_choice"*, *"weight_choice"*, *"influent_threshold"*, *"power"*, *"max_weight"*, *"comparison"*}:

4.4.1 Fixed Hyper-parameters

Two parameters are not tuned: ***num_mod***=*153* and ***split***=*0.67*, respectively the number of models involved in the ensemble and the fraction of the whole dataset used to build the training set. The final set of models involves variations of VSLNet and VSLBase models and their last three epochs.

4.4.2 Metrics Hyper-parameters

The first relevant set of hyper-parameters is the one affecting the metrics, from where weights are computed to determine the hierarchy of models. Namely: "***rank***", "***inter***", "***miou_choice***" and "***weight_choice***".

"*rank*" and "*inter*" refer to the metric "R@n, IoU= μ", where the latter represents the IoU threshold and the threshold a model's prediction must meet to vote in the voting schemes. While "*miou_choice*" represents the selected "mIoU" measure.

"*weight_choice*" is used to indicate whether to use any "mIoU"-measure or "R@n, IoU = μ" as a weighting measure.

4.4.3 Model Relevance Hyper-parameters

Another set of crucial hyper-parameters is the one related to the model's importance, namely: ***influent_threshold*** $\in \{0.67, 0.8\}$ and ***power*** $\in \{1, 10, 100\}$.

Where "*influent_threshold*" represents the fraction of the best performance (i.e., maximum weight) above which a model is considered influential in the voting process. Whilst "*power*" represents a standard hyperparameter that allows increasing or decreasing the contribution of the best models; the higher the power, the greater the contribution of the best models.

4.4.4 Voting Scheme Hyper-parameter

The hyperparameter determining the voting scheme is: "***max_weight***", where "*max_weight* = True" indicates that the voting scheme is prior, posterior otherwise.

4.4.5 Comparison Hyper-parameter

Lastly, there is the hyperparameter to decide whom to compare the ensemble model with: "***comparison***". If "*comparison* = best", then an "output.txt" file shows the ensemble model compared with best-performing on training data, which represents what needs to be surpassed for the method to be considered relevant. If "*comparison* = all", the ensemble model is compared with the best-performing model on the test data.

4.5. Results

4.5.1 Baseline Training

The performances obtained from the different configurations trained with the hyper-parameters shown in Table 2 are now compared with the official benchmark provided by [10]. Figure 5 shows that the models trained were able to beat the proposed benchmark. Since VSLNet architecture with Bert and EgoVLP has proven to be the best one, it's the configuration that has been fine-tuned in order to enhance the performances above the baseline ones.

It's also remarkable to notice that egoVLP features outperform omnivore ones: this could be due to the fact that the first model was specifically trained over egocentric videos, making it more precise and rich compared to the other one, which follows a wider and general approach. Moreover, BERT produces far better results than GloVe showing that the impact of the word encoder isn't at all negligible.

Finally, employing VSLBase results in a fall of performances even though this deviation is the one that has the minor impact compared to the two others. The reasons that cause this change in the results have been further analysed: Figure 6 evaluates the error measured as

Architecture	Features	Word Encoder	IoU = 0.3 (%) r@1	IoU = 0.3 (%) r@5	IoU = 0.5 (%) r@1	IoU = 0.5 (%) r@5	mIoU
VSLNet	Omnivore	BERT	6.48	14.04	3.72	8.67	5.04
		GloVe	3.05	8.34	1.21	4.34	2.73
	EgoVLP	BERT	**8.03**	**15.82**	**4.75**	**10.12**	**6.15**
		GloVe	3.79	9.55	2.17	5.83	3.37
VSLBase	Omnivore	BERT	6.61	13.37	3.59	8.60	5.13
		GloVe	3.30	8.80	1.99	5.52	3.17
	EgoVLP	BERT	7.18	14.74	4.26	9.65	5.66
		GloVe	3.87	8.78	2.14	5.58	3.35
VSLNet Benchmark			5.45	10.74	3.12	6.63	-

Figure 5. Results obtained for the baseline model

the difference between the length of the predicted interval and the one of the ground truth one (positive values mean bigger predicted intervals and negative values indicate smaller ones) for both VSLNet and VSLBase best architectures, it's clear that VSLBase slightly tends to predict longer time intervals resulting in a lower IoU if compared to VSLNet. In terms of running times, no substantial changes are reported when varying the architectures.

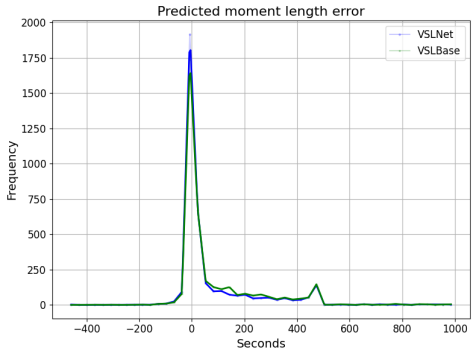


Figure 6. Moment length errors in seconds between ground truths and results predicted by VSLBase and VSLNet

4.5.2 Parameters influence exploration

Once the baseline model has been chosen, an exploration of the impact of the parameters is proposed. Every instance is trained for 20 epochs with different values for the parameters before mentioned.

The initial learning rate resulted in one of the most influential ones and its impact will be analysed by keeping all other parameters as in Table 2 besides the initial learning rate and the number of epochs. Figure 7 shows that a learning rate that is too low slows down the learning process but an initial value that it's too high has a terrible impact on performances, resulting in a sensible drop in terms of mIoU. The metric used here to report the general behaviour was only the mIoU as in this case there's a high correlation between this metric and "R@n, IoU = μ" (i.e. if a model is better in terms of one of them, then it is better also in terms of the other). Another aspect whose influence has been further analysed is the

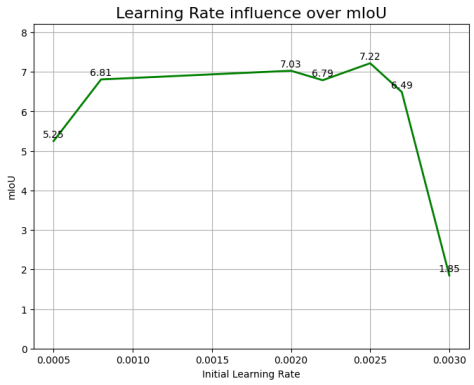
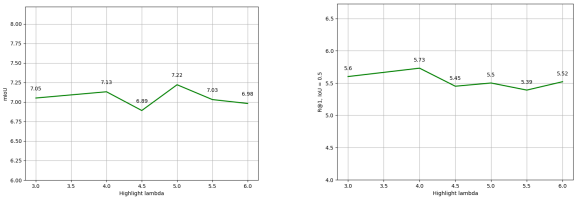


Figure 7. mIoU results as the initial learning rate varies

highlight lambda: all models share the same parameters apart from it, which are the ones shown before. Figures 8 show that in this case, the two metrics yield different results as the model having the highest mIoU doesn't coincide with the best one in terms of "R@n, IoU = μ".



(a) mIoU (b) R@1, IoU = 0.5

Figure 8. Performance trend as highlight parameter varies

After this exploration, two models have performed significantly better than the others:

- Model 1: parameters as in 2, lr = 0.0025, $\lambda = 5.0$
- Model 2: parameters as in 2, lr = 0.0025, $\lambda = 4.0$

The results obtained are provided in Figure 9.

4.5.3 Ensemble

Figure 10 shows three blocks, each one related to a different weighting scheme implied by, respectively, the

	IoU = 0.3 (%)		IoU = 0.5 (%)		mIoU
	r@1	r@5	r@1	r@5	
Model 1	**9.73**	**18.51**	5.50	12.13	**7.22**
Model 2	9.71	17.97	**5.73**	**12.16**	7.13

Figure 9. Best results obtained

	IoU = 0.3 (%)		IoU = 0.5 (%)		mIoU
	r@1	r@5	r@1	r@5	
Model 1	9.70	17.76	5.56	11.27	7.28
Ensemble A	10.33	17.53	6.18	10.41	7.60
Ensemble B	11.03	15.34	**6.96**	9.78	8.16
Model 2	9.55	**17.92**	5.79	**11.42**	7.18
Ensemble A	10.33	17.68	6.10	10.80	7.57
Ensemble B	11.11	15.34	6.81	9.78	8.17
Model 3	9.55	**17.92**	5.79	**11.42**	7.18
Ensemble A	10.41	17.61	6.03	10.49	7.57
Ensemble B	**11.35**	15.57	6.89	9.78	**8.18**

Figure 10. Comparison between the supposed best model and the ensemble ones

measures "mIoU", "mIoU_1" and "R@1 IoU $= \mu$". Each block contains the evaluation of three models: the first one, indicated with "Model", represents the evaluation on the test set of the best-performing model on the training set (also called supposed best). The second row, specified with "Ensemble A", is the evaluation of the best ensemble model found in the grid-search with the "Prior" voting scheme. Lastly, there is the evaluation of "Ensemble B", again the best model found, this time using the "Posterior" voting scheme.

The best-settings found for the ensembles are:
- mIoU:
Ensemble A: (*influent_threshold*=0.67, *power*=1)
Ensemble B: (*influent_threshold*=0.8, *power*=10)
- mIoU_1:
Ensemble A: (*influent_threshold*=0.8, *power*=1)
Ensemble B: (*influent_threshold*=0.8, *power*=10)
- R@1, IoU$= \mu \in 0.3, 0.5$:
Ensemble A:(*influent_threshold*=0.8, *power*=1, μ=*0.3*)
Ensemble B: (*influent_threshold*=0.8, *power*=1, μ=*0.3*)

Results are self-explanatory, the ensemble outperforms the supposed-best model under every measure, except for the "R@5, IoU$= \mu$", which has some drawbacks itself, since it might tend to reward models with sparse prediction throughout the clip, and less confidence on the first interval. Thus, the ensemble provides more accurate predictions of the first interval picked, which might show consistency with the other four predictions.

5. Conclusions

This report addresses the problem of NLVL as a span-based QA task. A baseline model is obtained and compared with the Ego4D NLQ official benchmark by implementing different architectures, testing two backbone word features extraction as well as employing different video pre-extracted features. Leveraging the results of this NLVL method allowed to obtain a textual answer by using the predicted interval and the language query as inputs for a VLM.

Furthermore, an analysis of how the parameters affect the performances on the NLVL task is performed in order to enhance the results previously obtained, as it also provides a deeper understanding of how models actually can learn and improve over this type of challenge.

Finally, an ensemble method is adopted to build a solid predictor, that exploits the computational power spent on training multiple models with different parameters, and provides significantly better predictions on single intervals. This innovative proposal concludes the report.

References

[1] https://github.com/ego4d/episodic-memory. 5

[2] Damen Dima et al. "rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100.". *International Journal of Computer Vision*, pages 1–23, 2022. 1

[3] Singh M. Ravi N. Van Der Maaten L. Joulin A. Misra I. Girdhar, R. Omnivore: A single model for many visual modalities. *roceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16102–16112, 2022. 2, 3

[4] Cordelia Schmid Ali Farhadi Gunnar A Sigurdsson, Abhinav Gupta and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018. 1

[5] Zhenheng Yang Jiyang Gao, Chen Sun and Ramakant Nevatia. Tall: Temporal activity localization via language query. *In IEEE International Conference on Computer Vision*, pages 5277–5285, 2017. 4

[6] Eugene Byrne Zachary Chavis Antonino Furnari Rohit Girdhar Jackson Hamburger Hao Jiang Miao Liu Xingyu Liu et al Kristen Grauman, Andrew Westbury. Ego4d: Around the world in 3,000 hours of egocentric video. 2022. 1

[7] Kevin Qinghong Lin, et al. Egocentric video-language pretraining. *Advances in Neural Information Processing Systems 35*, pages 7575–7586, 2022. 2, 3

[8] Zhu B. Ye Y. Ning M. Jin P. Yuan L Lin, B. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023. 3

[9] Tao Mei Yitian Yuan and Wenwu Zhu. To find where you talk: Temporal sentence localization in video with attention based location regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9159–9166, 2019. 4

[10] Hao Zhang, et al. Span-based localizing network for natural language video localization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. 2, 4, 5, 6