# Towards Parameter-Efficient Integration of Pre-Trained Language Models In Temporal Video Grounding

**Erica K. Shimomoto[1], Edison Marrese-Taylor[1], Hiroya Takamura[1], Ichiro Kobayashi[1,2], Hideki Nakayama[1,3], Yusuke Miyao[1,3]**
National Institute of Advanced Industrial Science and Technology[1]
Ochanomizu University[2], The University of Tokyo[3]
{kidoshimomoto.e,edison.marrese,takamura.hiroya}@aist.go.jp
koba@is.ocha.ac.jp, nakayama@ci.i.u-tokyo.ac.jp, yusuke@is.s.u-tokyo.ac.jp

## Abstract

This paper explores the task of Temporal Video Grounding (TVG) where, given an untrimmed video and a natural language sentence query, the goal is to recognize and determine temporal boundaries of action instances in the video described by the query. Recent works tackled this task by improving query inputs with large pre-trained language models (PLM) at the cost of more expensive training. However, the effects of this integration are unclear, as these works also propose improvements in the visual inputs. Therefore, this paper studies the effects of PLMs in TVG and assesses the applicability of parameter-efficient training with NLP adapters. We couple popular PLMs with a selection of existing approaches and test different adapters to reduce the impact of the additional parameters. Our results on three challenging datasets show that, without changing the visual inputs, TVG models greatly benefited from the PLM integration and fine-tuning, stressing the importance of sentence query representation in this task. Furthermore, NLP adapters were an effective alternative to full fine-tuning, even though they were not tailored to our task, allowing PLM integration in larger TVG models and delivering results comparable to SOTA models. Finally, our results shed light on which adapters work best in different scenarios.

## 1 Introduction

Temporal Video Grounding (TVG) is a fundamental task in Computer Vision (CV), where the goal is to have models recognize and determine temporal boundaries of action instances in videos (Shou et al., 2016; Gu et al., 2018; Girdhar et al., 2019) using queries provided in natural language (Gao et al., 2017; Hendricks et al., 2017). Over the past few years, interest in this task has grown substantially due to its complexity and potential applications, which has led to the release of several models that either use propose-and-rank techniques or directly predict the starting and ending temporal locations (Ghosh et al., 2019; Rodriguez et al., 2020).

Following trends in other vision-and-language (V&L) tasks, some of the latest models combine vision encoders and pre-trained language models (PLM). We find works that directly encode the query using PLMs (Nawaz et al., 2022; Wang et al., 2022), which are later fine-tuned with the rest of the architecture, or that try to project both the query and video into the same embedding space using a Transformer (Zhang et al., 2021a).

Despite the performance improvements, it is difficult to isolate the effects of the improved language representations, as these works also propose new video-language matching approaches or use different video encoders. Another drawback is their computational cost for training, as parameter counts grow substantially once PLMs are incorporated.

To address this problem, several parameter-efficient training methods have been recently proposed for both Natural Language Processing (NLP) (Karimi Mahabadi et al., 2021b) and CV models (Rebuffi et al., 2017). Among these approaches, adapters (Houlsby et al., 2019; Bapna and Firat, 2019) and their variations have been particularly effective[1], as they lead to performance as high as fine-tuning while training only a small set of parameters. Adapters have been successfully combined with vision models for several tasks (Kim et al., 2021; Zhou et al., 2022), showing that using a few parameters to learn to fuse vision and language representations without losing performance is possible. However, we note that efforts so far have focused only on image (Zhang et al., 2021b) and video (Pan et al., 2022) classification tasks or on leveraging pre-trained generative models, e.g., by re-casting existing vision-and-language tasks as language generation (Sung et al., 2022). In contrast,

[1]While the term "adapter" is often used to refer to the original adapter proposed by Houlsby et al. (2019), we use it to refer to any efficient fine-tuning method in this work.



*Findings of the Association for Computational Linguistics: ACL 2023*, pages 13101–13123
July 9-14, 2023 

as the training signal in TVG comes from the visual modality, we cannot cast it as language generation.

Therefore, this paper studies the effects of large PLMs in the TVG task and investigates the applicability of NLP adapters for a parameter-efficient integration. We couple popular PLM models with a selection of previous works, allowing us to isolate and understand their effects on performance. Concretely, we analyze ExCL (Ghosh et al., 2019), TMLGA (Rodriguez et al., 2020) and DORi (Rodriguez-Opazo et al., 2021), three proposal-free TVG models with different levels of complexity. Moreover, we also benchmark several parameter-efficient training alternatives based on adapters.

We conduct thorough experiments on three challenging datasets, Charades-STA (Gao et al., 2017), ActivityNet Captions (Krishna et al., 2017) and YouCookII (Zhou et al., 2018b,a), covering videos and queries with varying lengths of different activities. Concretely, we seek to answer the following research questions: **RQ1**: Does incorporating a PLM improve the performance of existing TVG models?; **RQ2**: Are adapters an alternative to full fine-tuning of the PLM parameters within TVG?; **RQ3**: Is there an adapter that works best for TVG?; **RQ4**: What is the impact of different PLMs?; **RQ5**: How does the combination of existing TVG models with PLMs trained with adapters perform against state-of-the-art models?

Our results offer concrete answers to these questions, helping us clarify the role of PLMs in the TVG task and quantify how much they can improve the existing model's grounding capabilities. They suggest that, by only changing the query sentence representation using PLMs, TVG models can greatly improve performance, especially when PLMs are fine-tuned, stressing the importance of the text query representation in this task.

Our contributions can be summarized as follows: (1) We quantify the impact of PLMs in TVG models, (2) We perform the first work on benchmarking different types of adapters on the TVG task, shedding light on which adapters work best for each case, and (3) We offer an empirical demonstration that adapters can reach or surpass the performance of full fine-tuning while updating only $\sim 10\%$ of the parameters in our task. The code to reproduce our experiments is available at `github.com/ericashimomoto/parameter-efficient-tvg`.

## 2 Related Work

**Temporal Video Grounding:** Work on this task can be divided into two main approaches. On the one hand, we find techniques based on proposal generation, where the idea is to, given a query, output a set of candidate clips which could later be ranked (Liu et al., 2018; Ge et al., 2019). Further research has mostly focused on reducing the number of proposals by producing query-guided or query-dependent approaches (Chen et al., 2018; Chen and Jiang, 2019; Xu et al., 2019), or on creating maps that can cover diverse video moments with different lengths (Zhang et al., 2021c). Zhang et al. (2021a) adopted a Transformer-based multi-modal model (MSAT) which is pre-trained for this setting. More recently, models have incorporated contrastive losses to improve performance further. This is the case of both CPL (Zheng et al., 2022) and MNM (Wang et al., 2022), which also incorporate Transformer-based components in their pipelines.

The second line of approaches has instead proposed to directly predict the start and end locations through the video span (ExCL; Ghosh et al., 2019). Work on this line of research has focused on improving the performance by modelling label uncertainty (TMLGA; Rodriguez et al., 2020), adding spatial features (DORi; Rodriguez-Opazo et al., 2021) or improving the text-to-video matching strategies (Mun et al., 2020; Zeng et al., 2020). For example, CPN (Zhao et al., 2021) adopted an ad-hoc graph-based technique. Other approaches have focused on exploiting local and global features for better performance like Mun et al. (2020). CP-Net (Li et al., 2021) recently proposed a pyramid-like approach where the model progressively replenishes the temporal contexts and refines the location of the queried activity by enlarging the temporal receptive fields. Finally, two recent approaches, VSLNet (Zhang et al., 2020) and BCPN, (Nawaz et al., 2022) have proposed to cast the task as visual question answering.

On top of these lines, recently, an effort has been made to solve a variant of the task named spatio-temporal video grounding. In this case, besides predicting when the moment starts and ends, the model should also identify where the action described by the textual query occurs in the frames. Works on this task heavily rely on the transformers architecture (Yang et al., 2022a), with significant effort in eliminating the need for any pre-trained object detectors (Su et al., 2021; Jin et al., 2022).

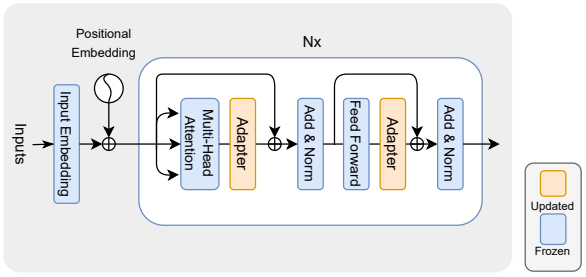



Figure 1: Illustration of the traditional bottleneck adapter proposed by Houlsby et al. (2019). The adapter layers are introduced after the multi-head attention and feed-forward layers. Orange color refers to trainable parameters, and blue color refers to frozen ones.

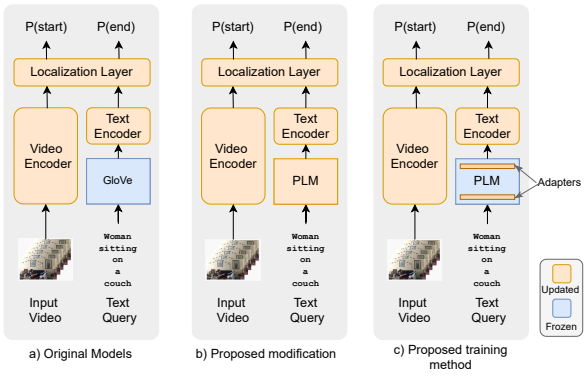



Figure 2: Illustration of our proposed approach to incorporate pre-trained language encoders and adapters into existing Temporal Video Grounding pipelines. Orange color refers to trainable parameters, and blue color refers to frozen ones.

While closely related to our task, due to the addition of the spatial dimension, this task naturally emphasizes the role of the visual modality in the grounding, further deviating from our language-driven approach. Therefore, our study does not consider models tailored for this task.

**Parameter-efficient model training:** As machine learning models continue to grow, updating their parameters efficiently is becoming increasingly important. One key idea has been to only update newly-added parameters (Rebuffi et al., 2017, 2018). With the advent of large pre-trained Transformer-based models in NLP, this idea has led to the development of adapters (Houlsby et al., 2019): sub-networks with few parameters that are inserted after every attention and feed-forward layer in a given model, as illustrated in Figure 1. We also find a variety of prompt-based approaches, which add trainable parameters into the model inputs (Li and Liang, 2021; Lester et al., 2021; Gu et al., 2022). Alternative techniques, such as sparsely updating a small number of parameters of the model (Ben Zaken et al., 2022; Guo et al., 2021; Sung et al., 2021), or low-rank factorization for the weights to be updated (Mahabadi et al., 2021; Karimi Mahabadi et al., 2021a; Hu et al., 2022) have also been proposed recently. Finally, He et al. (2022a); Mao et al. (2022) combined some of these techniques to propose a unified parameter-efficient training framework. Though we focus on NLP adapters, we deviate from previous work as our approach incorporates the visual modality.

Only updating newly-added parameters has also been proposed in CV, with some work predating the advent of Transformers (Rebuffi et al., 2017, 2018). More recently, we find works combining pre-trained language models with multi-modal inputs. For example, Tsimpoukelli et al. (2021) trained a vision encoder to represent each image as a sequence of continuous embeddings, such that a frozen language model prompted with this prefix generates the appropriate image caption. Yang et al. (2022b) showed that it is possible to perform zero-shot video question answering by leveraging frozen bidirectional language models. More recently, Sung et al. (2022) cast multiple V&L tasks as text generation, combining NLP adapters with pre-trained encoder-decoders, such as BART (Lewis et al., 2020), with existing image encoders, such as CLIP (Radford et al., 2021). The latter has also lately been the target of several studies that extend parameter-efficient techniques for CV (Kim et al., 2021; Zhang et al., 2021b; Zhou et al., 2022).

Though our approach follows a similar trend, our interest lies in a grounding task where the training signal comes from the visual modality, which keeps us from casting our task as language generation.

## 3 Proposed Approach

Consider a video $V \in \mathcal{V}$, represented as a sequence of frames such that $V = \{v_t\}$ with $t = 1, \ldots, l$. Each video in $\mathcal{V}$ is annotated with a natural language passage $S \in \mathcal{S}$, where $S$ is a sequence of words $S = \{s_j\}$ with $j = 1, \ldots, m$, which describes what is happening at a certain period of time in the video. This interval is formally defined by $t_s$ and $t_e$, the starting and ending points of the annotations in time, respectively. The goal of the temporal video grounding task is to predict $t_s$ and $t_e$ given the video $V$ and the text query $S$.

### 3.1 Temporal Video Grounding Models

For this study, we focused on proposal-free models, which generally offer better performance, and we were careful only to consider works that used word embeddings. Furthermore, we only considered models which have their implementation available. Concretely, we selected ExCL, TMLGA, and DORi. ExCL was the first proposal-free model for TVG; TMLGA improved on it by handling video annotation uncertainty, and later DORi incorporated spatial features.

As shown in Figure 2 (a), these models can be summarized into three main parts: a sentence encoder, a video encoder, and a localization module, which combines information from both modalities and predicts the start and end points of the segment in the video described by the query. While these models heavily explore information from the video input, they make relatively simple use of the sentence query, mainly processing pre-trained word embeddings, such as GloVe (Pennington et al., 2014), through a recurrent neural network.

To better understand the role of improved language representations on the performance in the TVG task, we study the effects of PLMs by incorporating them as-is into a selection of existing models from the literature, therefore effectively isolating their impact on the performance across a range of settings. We tested several ways to incorporate PLMs, such as entirely replacing the text encoder block with the PLM, where most failed to deliver any performance improvement, and, therefore, decided to replace only the word embeddings, as shown in Figure 2 (b).

### 3.2 Adapters

A concern when integrating large PLMs with existing TVG models is the alarming number of parameters added. While some TVG models are pretty small, recent models are exponentially increasing in size. For example, DORi has about 10M parameters, more than double than TMLGA (about 4M). Combining these models even with reasonably-sized PLMs, such as BERT (Devlin et al., 2019), increases the number of trainable parameters to over 120M. Also, the more sophisticated visual features are used, the larger the training data becomes. For example, the visual features for TMLGA are about 2.1MB per video in the ActivityNet dataset, while for DORi, they are about 82MB.

To alleviate this issue, we also investigate several parameter-efficient training alternatives based on adapters to incorporate these PLMs into the existing model pipelines with reduced computational cost, as shown in Figure 2 (c).

For our experiments, we adopt a large selection of adapters following previous work (Sung et al., 2022), including bottleneck adapters, such as the ones proposed by Houlsby et al. (2019) (HOULSBY), Pfeiffer et al. (2020b) (PFEIFFER), and He et al. (2022a) (PARALLEL); Invertible adapters (INVERSE) (Pfeiffer et al., 2020b), Prefix Tuning (PREFIX) (Li and Liang, 2021), Compacter (COMPACTER) (Karimi Mahabadi et al., 2021a), and LoRA (LORA) (Hu et al., 2022).

We note that some of the adapters we consider in our study were designed for specific purposes in NLP. Our decision to still include such adapters in our experimental framework is motivated by their relative success in other vision-and-language tasks (He et al., 2022c; Kim et al., 2021; Sung et al., 2022). Moreover, as our approach differs from existing work in this context, we were interested in offering empirical evidence to further understand these adapters' role in multi-modal scenarios.

## 4 Experimental Framework

### 4.1 Datasets

**Charades-STA:** Built upon the Charades dataset (Sigurdsson et al., 2016), it provides time-based annotations using a pre-defined set of activity classes and general video descriptions. We use the predefined train and test sets containing 12,408 and 3,720 moment-query pairs, respectively. Videos are 31s long on average and have a maximum duration of 194s, with 2.4 moments on average, each being 8.2s long on average and described using 7.2 words on average.

**ActivityNet Captions:** Introduced by Krishna et al. (2017) and initially constructed for dense video captioning, it consists of 20k YouTube videos with an average length of 120s and a maximum duration of 755s. The videos contain 3.65 temporally localized time intervals and sentence descriptions on average, where descriptions have on average 13.48 words. Following previous works, we report the performance on the combined validation sets.

**YouCookII:** It consists of 2,000 long untrimmed videos from 89 cooking recipes obtained from YouTube by Zhou et al. (2018b,a). Each step for cooking these dishes was annotated with temporal boundaries and aligned with the corresponding

section of the recipe. The average video length is 316s and a maximum duration of 755s. Regarding relevant moment segments, each video has 7.73 moments on average, with each segment being 19.63s long and described using 8.7 words on average.

### 4.2 Implementation Details

**Temporal Video Grounding Models:** Our implementation for TMLGA and DORi is built on top of the original code released by the authors, while we use our own implementation of ExCL. To represent the video input, we follow the original implementations as close as possible. For TMLGA and DORi, we use the I3D features released with the respective papers. For ExCL, we use the features released by the DORi paper, extracted at 25 fps instead of the original 5 fps[2]. Specifically for DORi, we also use the spatial features released with the paper.

**Pre-trained Language Models:** We combine our selected models with pre-trained BERT, RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2022b), with the implementations provided by the HuggingFace library (Wolf et al., 2020). Furthermore, we used "bert-base-uncased", "roberta-base", and "deberta-base" pre-trained models.

**Adapters:** We use the implementation provided by the adapter-transformers library (Pfeiffer et al., 2020a), with default configurations. In particular, for the Invertible adapters (INVERSE), we only tested the "PfeifferInvConfig", the original configuration proposed in the paper (Pfeiffer et al., 2020b).

**Training:** Our experiments were performed on a 40-GB NVIDIA A100 GPU. Models were trained using ADAM (Kingma and Ba, 2020) with a step-based learning rate scheduler. When fine-tuning the PLMs, we used a scheduler with linear warmup for the PLM parameters, while keeping the learning rate fixed for the rest of the parameters. For more details on hyper-parameters, we refer the readers to the Appendices A and B. The evaluation follows previous work, based on two widely used metrics (Gao et al., 2017), namely the Recall at various thresholds of the temporal Intersection over Union (tIoU or $R@\alpha$) measuring the percentage of predictions that have tIoU with ground truth larger than certain $\alpha$, and the mean averaged tIoU (mIoU). We use $\alpha$ threshold values of 0.3, 0.5 and 0.7.

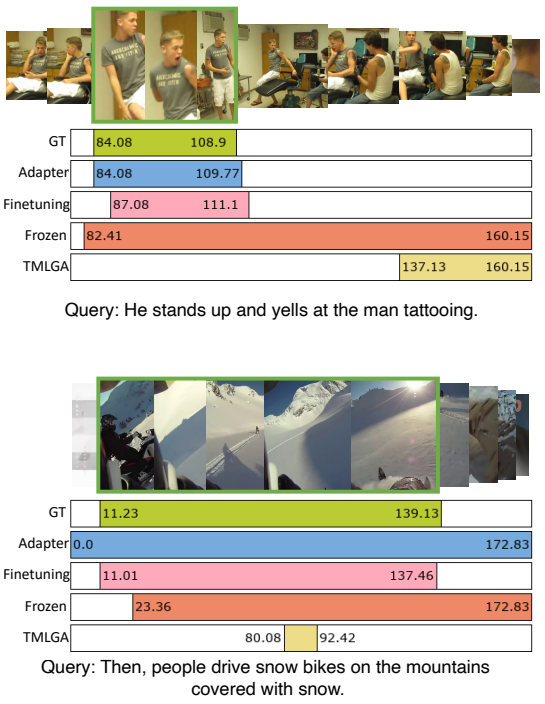



Figure 3: Examples of success (top) and failure (bottom) of TMLGA with BERT on ActivityNet.

## 5 Results and Discussion

**RQ1: Effect of adding BERT to existing models** To investigate this matter, we replace the non-contextualized word embeddings in our selected TVG models with BERT, which can be regarded as the current most widely-studied PLM (Rogers et al., 2020; Yang et al., 2020; Chen et al., 2020; Li et al., 2020). We compare the original model performance with the performance when fine-tuning the PLM along with the TVG model training (fine-tuning), and when freezing the PLM (❄), training only the parameters of the TVG model. To ensure our implementations were correct, we also tested the original models (ours), achieving performance close to the reported in their respective papers.

Table 1 shows the results of this combination. When introducing BERT to the models, we can see that full model fine-tuning leads to an average improvement of 1.38% and 1.24% in mIoU for ExCL and TMLGA, respectively. This result shows that the chosen TVG models can benefit from using PLMs. However, BERT adds over 100M parameters to be tuned. Such an increase is troublesome as we ran out of memory when trying to fine-tune

[2]In practice, this difference in fps is not an issue, as Rodriguez-Opazo et al. (2021) has shown that extraction at 25fps leads to better performance

| Method | Params. | Charades-STA | | | | ActivityNet | | | | YouCookII | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU |
| ExCL† (orig.) | 6.9M | 65.10 | 44.10 | 22.60 | - | - | - | - | - | 44.20 | 28.00 | 14.60 | - |
| ExCL (ours) | 6.9M | 62.28 | 39.74 | 22.53 | 42.28 | 55.49 | 39.33 | 23.04 | 40.32 | 26.58 | 15.72 | 8.19 | 18.99 |
| + BERT ❄ | 6.9M | 62.93 | 38.44 | 22.23 | 42.38 | 57.21 | 39.66 | 23.79 | 41.45 | 26.63 | 16.15 | 8.51 | 18.87 |
| + Adapter | 7.2M-16.8M | 61.59 | 37.15 | 21.51 | 41.06 | 59.35 | 41.27 | 24.86 | 42.83 | 28.47 | 17.75 | 9.02 | 19.89 |
| + Fine-tuning | 116M | 61.75 | 38.36 | 23.44 | 42.00 | 59.10 | 41.83 | 25.42 | 42.36 | 28.18 | 16.84 | 9.02 | 20.08 |
| TMLGA (orig.) | 4.7M | 67.53 | 52.02 | 33.74 | 48.22 | 51.28 | 33.04 | 19.26 | 37.78 | 33.48 | 20.65 | 10.94 | 23.07 |
| TMLGA (ours) | 4.7M | 69.49 | 49.97 | 32.72 | 48.29 | 50.84 | 31.13 | 17.86 | 36.90 | 34.42 | 21.99 | 10.94 | 23.63 |
| + BERT ❄ | 4.7M | 70.08 | 49.92 | 31.42 | 48.34 | 52.10 | 32.57 | 18.64 | 37.63 | 34.77 | 23.05 | 12.49 | 24.42 |
| + Adapter | 5.6M - 14.6M | 71.40 | 52.53 | 33.82 | 49.57 | 53.98 | 35.20 | 20.43 | 38.88 | 36.08 | 22.77 | 12.49 | 25.19 |
| + Fine-tuning | 114M | 71.02 | 52.53 | 33.52 | 49.80 | 53.59 | 34.05 | 19.51 | 37.92 | 35.34 | 21.85 | 11.63 | 24.82 |
| DORi (orig.) | 10.4M | **72.72** | **59.65** | 40.56 | 53.28 | 57.89 | 41.49 | 26.41 | 42.78 | 43.36 | 30.47 | 18.24 | 30.46 |
| DORi (ours.) | 10.4M | 72.26 | 57.18 | 40.62 | 53.01 | 57.38 | 40.00 | 24.84 | 41.97 | 43.33 | 29.15 | 17.61 | 30.17 |
| + BERT ❄ | 10.4M | 71.83 | 57.15 | 39.22 | 52.49 | 58.86 | 40.86 | 25.50 | 42.97 | 42.27 | 29.90 | 18.38 | 29.92 |
| + Adapter | 11.6M - 20.3M | 72.50 | 58.63 | **40.97** | **53.29** | **60.81** | **43.49** | **27.86** | **44.55** | **46.79** | **32.56** | **19.87** | **32.48** |

Table 1: Overview of our results combining BERT and adapters with our selected prior work. Underlined results indicate the best performance within the method and dataset combination, while results in bold indicate the best performance within the dataset.

BERT along with DORi.

One alternative to save on this computational cost is to freeze the PLM when training the TVG model. We can see that this strategy leads to an overall improvement in mIoU of $0.40\%$, $0.52\%$, and $0.08\%$ for ExCL, TMLGA, and DORi, respectively, when compared to the original model performance. This improvement is substantially smaller than when fine-tuning the PLM, with cases where using the frozen BERT leads to worse results than when using GloVe, such as with ExCL and DORi on YouCookII. Furthermore, we can see that the models benefited the least from the frozen PLM when tested on the Charades-STA dataset. This result could be due to queries in Charades-STA being less complex, so the word embeddings could already be enough to perform well. Nevertheless, the overall results indicate that fine-tuning is essential to getting the full potential of the PLM in our task.

**RQ2: Adapters as an alternative to PLM fine-tuning** To the best of our knowledge, there is no evidence to suggest whether adapters could bring benefits to the TVG task similar to what has been shown in other NLP tasks. Therefore, we seek to investigate if using adapters can be an effective alternative to full fine-tuning of the PLM models in TVG. We tested the adapters mentioned in Section 3.2 for all three TVG models with BERT.

Our best results are shown in Table 1 (Adapter). For ExCL, the best adapters were PFEIFFER, PFEIFFER, and LORA, for Charades-STA, ActivityNet and YouCookII, respectively; For TMLGA, the best were PREFIX, PFEIFFER, and HOULSBY; and for DORi, INVERSE, PREFIX, and HOULSBY. Further results can be found in Appendix A. We also provide the visualization of success and failure examples in Figure 3 for the combination of TMLGA with BERT on ActivityNet, and refer the readers to the Appendix C for more visualizations.

Our results show that using adapters led to an overall improvement in mIoU of $0.70\%$, $1.71\%$ and $1.72\%$ for ExCL, TMLGA, and DORi, respectively, over the models' original performance. While the improvement from the adapters for ExCL is smaller than when doing full fine-tuning, adapters led to better performance with TMLGA. More importantly, adapters allowed a significant performance improvement for DORi, as training with them requires updating only 16% of the parameters required for full fine-tuning. Furthermore, we can see that in some cases, using adapters leads to better performance than full fine-tuning, such as with ExCL and TMLGA on ActivityNet Captions.

In summary, these results indicate that despite not being tailored for the task of TVG, the adapters covered in this work can be an efficient alternative to the full fine-tuning of PLM models.

**RQ3: Choice of adapter for TVG** We were then naturally interested in studying whether there is a specific type of adapter that works best to solve our task. Therefore, we ranked the performance of each adapter for each dataset and model, focus-

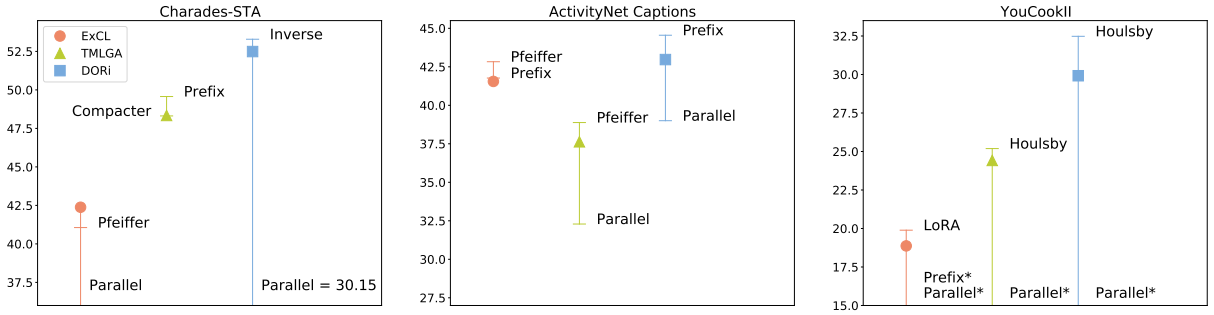



Figure 4: Best and worst performing adapters with BERT. The y-axis represents the performance in terms of mIoU. The circles, triangles and squares indicate the performance with frozen BERT, while the horizontal ticks show the performance of the models with the corresponding adapter. The * indicates cases where the models could not learn properly with the adapter.

ing on cases that deliver the best and the worst performance.

The results of our analysis are summarized in Figure 4. We can see that no single adapter can consistently offer the best performance. However, it is possible to see that the bottleneck adapters, such as HOULSBY, PFEIFFER, and INVERSE, can deliver an overall better performance.

When controlling our results for each dataset, it is possible to see that the PREFIX and the INVERSE adapters worked well on Charades-STA, while the PFEIFFER and HOULSBY adapters worked best for ActivityNet and YouCookII, respectively. We surmise this indicates that the adapter choice is likely more related to the training data than to the choice of the model itself. Controlling by model also shows interesting, distinctive patterns. For example, while the PREFIX adapter worked well for TMLGA on Charades-STA and DORi on ActivityNet, it performed poorly with ExCL on ActivityNet and YouCookII.

Another general pattern we could identify is that both the COMPACTER and the PARALLEL adapters did not work well overall, with the latter struggling to deliver good performance in all cases. We think the number of additional parameters may play a role in this matter, as COMPACTER is the adapter with the smallest number of added parameters (0.06M) and it might be that too small to learn useful information for TVG.

Furthermore, the hyper-parameter search with the PREFIX and the PARALLEL adapters was substantially more challenging. In particular, with the PARALLEL adapter, we faced gradient explosions in many instances, as with DORi on Charades-STA, or the model converged to poor performance, as with all three models on YouCookII.

| Model | Charades-STA | | | ActivityNet | | |
|---|---|---|---|---|---|---|
| | R@0.5 | R@0.7 | mIoU | R@0.5 | R@0.7 | mIoU |
| TMLGA (ours) | 49.97 | 32.72 | 48.29 | 31.13 | 17.86 | 36.90 |
| + BERT ❄ | 49.92 | 31.42 | 48.34 | 32.57 | 18.64 | 37.63 |
| + PFEIFFER | 51.59 | 33.41 | 49.50 | 35.20 | 20.43 | 38.88 |
| + INVERSE | 52.77 | 34.49 | 49.33 | 33.76 | 19.93 | 37.93 |
| + Fine-tuning | 52.53 | 33.52 | 49.80 | 34.05 | 19.51 | 37.92 |
| + RoBERTa ❄ | 51.34 | 33.49 | 48.91 | 33.80 | 19.62 | 37.89 |
| + PFEIFFER | **53.84** | **34.78** | **49.91** | 35.27 | 20.26 | 38.77 |
| + INVERSE | 52.69 | 33.98 | 49.50 | 34.85 | 20.46 | 39.35 |
| + Fine-tuning | 53.15 | 33.33 | 49.77 | 33.21 | 19.69 | 38.47 |
| + DeBERTa ❄ | 52.53 | 33.49 | 49.32 | 33.94 | 20.22 | 38.72 |
| + PFEIFFER | 53.49 | 34.65 | 49.78 | 34.70 | 20.49 | 39.30 |
| + INVERSE | 52.58 | 33.95 | 49.66 | **35.45** | **20.66** | **39.71** |
| + Fine-tuning | 53.44 | 33.44 | 49.63 | 33.78 | 20.12 | 38.92 |

Table 2: Detailed results combining the TMLGA model with our three PLMs and adapters, tested on Charades-STA and ActivityNet Captions. Underlined results indicate the best performance within the model and dataset combination, while the results in bold indicate the best performance within the dataset.

Finally, we believe another factor is the location where each adapter is inserted in the PLM architecture. All of the methods tested adapt to specific parts of the transformer layer, except for the PARALLEL adapter, which adapts the whole transformer layer. Its consistent poor performance might indicate that adapting specific parts of the transformer layer is more beneficial for our task.

**RQ4: Impact of different PLMs** After the release of BERT, several pre-trained language encoder variations have been proposed. When tested on extensive NLP benchmarks, some of these models have proven to be able to consistently improve performance. In this context, we are interested in studying if such performance improvements also

translate to better performance in our task.

Thus, we study the performance of BERT, RoBERTa, and DeBERTa on two datasets, Charades-STA and ActivityNet, using TMLGA as a pivot. Our choice of TVG model is guided by our experiments with BERT, where TMLGA offered a good compromise in terms of performance improvements versus computational cost. Furthermore, our dataset selection is motivated by Charades-STA and ActivityNet having similar video contents but with different complexity queries, i.e., queries in Charades-STA are much simpler than in ActivityNet. We can verify this difference by observing the vocabulary size (748 vs. 9,744 tokens) and length of the queries (7.2 vs. 13.48 tokens per query).

The results are summarized in Table 2. For better readability, we only included two $\alpha$ bands. Further results can be seen in Appendix B. We can first see that for all three PLMs, using adapters led to an overall performance improvement compared to the frozen PLM. Moreover, we noticed that PFEIFFER and the INVERSE adapters seem to provide the best results in both datasets and all PLMs.

Moreover, while DeBERTa performed the best for ActivityNet, as expected from its performance in NLP downstream tasks, its best performance was similar to BERT for Charades-STA. We believe this result could be due to the simplicity of the queries in Charades-STA, which might not require all the additional information DeBERTa encodes. In addition, these results were obtained using the first version of DeBERTa, which uses the same type of tokenizer as BERT. We also tested the newer version of DeBERTa, which uses a sentencepiece-based tokenizer and incorporates other model improvements. However, this model performed much worse than the first version on our task showing that using sophisticated tokenizers does not necessarily improve results with simple sentence queries.

On the other hand, ActivityNet has longer and more complex queries and our results indicate that in such cases, our task might benefit from using better PLMs. Nevertheless, the best results were achieved when using adapters.

**RQ5: Comparison against state-of-the-art** We finally compare our best-performing models against a selection of approaches from previous work. We achieved our best performance for the Charades-STA dataset by using DORi with DeBERTa+PFEIFFER; and for the ActivityNet dataset, DORi with DeBERTa+INVERSE. Since the models used in this study are proposal-free, we mainly compare to proposal-free methods, i.e., VSLNet, CPN, CPNet and BCPN. Nevertheless, we are also interested in observing how our modifications perform against proposal-based methods, such as MS-2D-TAN (Zhang et al., 2021c), MSAT, CPL, and MNM. We note that out of the proposal-free methods, only BCPN uses a PLM (BERT). As for the proposal-based methods, only MS-2D-TAN does not use any form of Transformers in its architecture, while MNM is the only one to use a fine-tuned PLM (DistilBERT). Finally, while MSAT and CPL use Transformers in their architecture to encode visual and textual information, they do not use PLMs.

| Model | Charades-STA | | | ActivityNet | | |
|---|---|---|---|---|---|---|
| | R@0.5 | R@0.7 | mIoU | R@0.5 | R@0.7 | mIoU |
| Proposal-free | | | | | | |
| DORi (ours) | 57.18 | 40.62 | 53.01 | 40.00 | 24.84 | 41.97 |
| + DeBERTa ❄ | 58.17 | 40.94 | 52.73 | 41.65 | 25.82 | <u>43.64</u> |
| + Adapter | 58.39 | <u>41.61</u> | **53.34** | **45.63** | <u>28.74</u> | **45.70** |
| VSLNet | 54.19 | 35.22 | 50.02 | 43.22 | 26.16 | 43.19 |
| CPNet | <u>60.27</u> | 38.74 | 52.00 | 40.56 | 21.63 | 40.65 |
| CPN | 59.77 | 36.67 | <u>53.14</u> | <u>45.10</u> | 28.10 | **45.70** |
| BCPN | **61.77** | **43.91** | - | 44.53 | **30.11** | - |
| Proposal-based | | | | | | |
| MS-2D-TAN | **60.08** | **37.39** | - | 45.50 | 28.28 | - |
| MSAT | - | - | - | 48.02 | 31.78 | - |
| CPL | <u>49.24</u> | 22.39 | - | **55.73** | **31.37** | - |
| MNM | 47.31 | <u>27.28</u> | - | <u>48.59</u> | <u>29.26</u> | - |

Table 3: Comparison between the best performing TVG model with DeBERTa and current state-of-the-art methods in TVG. Results for all compared methods were taken from their respective papers. The best results for each combination of method type (i.e., proposal-free or proposal-based) and dataset are indicated in bold, while the second-best results are underlined.

The performance summary is shown in Table 3. For this analysis, we only consider two $\alpha$ bands for the thresholds as most proposal-based models do not report results at the $\alpha$ 0.3.

First, we can see that the best-performing methods are proposal-free, with BCPN achieving the best performance in most of the considered metrics. Furthermore, looking at the mean tIoU, we can see that DORi already performs well against the other methods. Replacing GloVe embeddings with DeBERTa and using adapters for training provides a significant performance boost, delivering results equivalent to state-of-the-art models.

Moreover, it is interesting to see that while

proposal-based methods such as CPL and MNM performed well against all methods on ActivityNet, they could not outperform methods without Transformers on the Charades-STA dataset. In contrast, DORi with DeBERTa and adapters achieved a more balanced performance among both datasets, showing a clear advantage against these methods on the Charades-STA dataset.

Therefore, our results show how TVG models can greatly benefit from adequately incorporating PLMs and making use of parameter-efficient techniques, performing well on datasets with different complexity levels in terms of queries, and achieving results comparable to state-of-the-art methods.

## 6 Conclusions

This paper studied the effects of PLMs in the TVG task and assessed the applicability of NLP parameter-efficient training alternatives based on adapters. We coupled BERT, RoBERTa, and DeBERTa, with a selection of previous TVG works, i.e., ExCL, TMLGA, and DORi, and tested different adapters to reduce the impact of the additional parameters. Our results showed that, by only changing the query representation using PLMs, TVG models can greatly benefit from such integration, especially when PLMs are fine-tuned, highlighting the importance of the query representation in this task. Moreover, we verified that adapters are an effective alternative to full fine-tuning, even though they were not tailored for our task. They saved on computational cost, allowing improvements for larger TVG models, such as DORi, and also delivered results comparable to SOTA models. Finally, we observed that while PARALLEL adapters struggled to learn in this task, bottleneck adapters such as HOULSBY and PFEIFFER performed across all tested TVG models and datasets.

## Limitations

In this work, we studied the effects of large pretrained models in the temporal video grounding task and investigated the applicability of NLP adapters for a parameter-efficient integration. While we believe our results show the efficacy of incorporating better language models in TVG models, it is important to note that we primarily focused on proposal-free TVG models and thus have no evidence to suggest such improvement would be observed in proposal-based models.

Furthermore, as our main goal was to investigate how the chosen models' performance varied when only changing the text encoding models, we compared state-of-the-art models using different visual features. While it would be interesting and insightful to check their performance when using the same features as our chosen models (i.e., I3D), such experiments are out of the scope of this study.

Moreover, although language adapters can be stacked before a task adapter for training on the task in a new language, we have only experimented with queries in English. It would be interesting to investigate if language adapters could be applied to TVG in different languages.

Finally, as for hardware requirements, our experiments were performed on a single 40-GB NVIDIA A100 GPU from a large cluster, and we spent about 400 USD on our experimental setup. While experiments with ExCL and TMLGA can be run on smaller GPUs with no significant increase in training time (i.e., we tested with a 16-GB NVIDIA V100 GPU), for DORi, due to the size of the input features and number of training parameters, we recommend using a GPU with at least 32GB of memory.

## Ethics Statement

This work does not present any direct ethical issues. Code for TMLGA and DORi were released by their respective authors. The datasets used to evaluate our proposed approach are open-access, and data characteristics relevant to our task were described in the experimental evaluation section. References for further information on each dataset were included in the paper.

## Acknowledgements

This paper is based on results obtained from the project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). For experiments, computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

## References

Ankur Bapna and Orhan Firat. 2019. Simple, Scalable Adaptation for Neural Machine Translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

*9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 162–171, Brussels, Belgium. Association for Computational Linguistics.

Shaoxiang Chen and Yu-Gang Jiang. 2019. Semantic proposal for activity localizaiton in videos via sentence query. *AAAI*.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. UNITER: UNiversal Image-TExt Representation Learning. In *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 104–120, Cham. Springer International Publishing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *ICCV*.

Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. 2019. Mac: Mining activity concepts for language-based temporal localization. In *WACV*.

Soham Ghosh, Anuva Agarwal, Zarana Parekh, and Alexander Hauptmann. 2019. ExCL: Extractive Clip Localization Using Natural Language Descriptions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. 2019. Video Action Transformer Network. In *CVPR*, pages 244–253.

Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. 2018. AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained Prompt Tuning for Few-shot Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. Association for Computational Linguistics.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2022b. DeBERTa: Decoding-Enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations*.

Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. 2022c. Parameter-efficient Fine-tuning for Vision Transformers.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Yang Jin, Zehuan Yuan, Yadong Mu, et al. 2022. Embracing consistency: A one-stage approach for spatio-temporal video grounding. *Advances in Neural Information Processing Systems*, 35:29192–29204.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 1022–1035. Curran Associates, Inc.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576, Online. Association for Computational Linguistics.

Konwoo Kim, Michael Laskin, Igor Mordatch, and Deepak Pathak. 2021. How to Adapt Your Large-Scale Vision-and-Language Model.

Diederik P. Kingma and Jimmy Ba. 2020. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-Captioning Events in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 706–715.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Kun Li, Dan Guo, and Meng Wang. 2021. Proposal-Free Video Grounding with Contextual Pyramid Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(3):1902–1910.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks. In *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 121–137, Cham. Springer International Publishing.

Meng Liu, Xiang Wang, Liqiang Nie, Xiangnan He, Baoquan Chen, and Tat-Seng Chua. 2018. Attentive moment retrieval in videos. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 15–24. ACM.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland. Association for Computational Linguistics.

Jonghwan Mun, Minsu Cho, and Bohyung Han. 2020. Local-Global Video-Text Interactions for Temporal Grounding. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10807–10816, Seattle, WA, USA. IEEE.

Hafiza Sadia Nawaz, Zhensheng Shi, Yanhai Gan, Amanuel Hirpa, Junyu Dong, and Haiyong Zheng. 2022. Temporal Moment Localization via Natural Language by Utilizing Video Question Answers as a Special Variant and Bypassing NLP for Corpora. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1.

Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. 2022. ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning for Action Recognition.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based

Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient Parametrization of Multi-Domain Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127.

Cristian Rodriguez, Edison Marrese-Taylor, Fatemeh Sadat Saleh, Hongdong Li, and Stephen Gould. 2020. Proposal-free Temporal Moment Localization of a Natural-Language Query in Video using Guided Attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2464–2473.

Cristian Rodriguez-Opazo, Edison Marrese-Taylor, Basura Fernando, Hongdong Li, and Stephen Gould. 2021. DORi: Discovering Object Relationships for Moment Localization of a Natural Language Query in a Video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1079–1088.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Zheng Shou, Dongang Wang, and Shih-Fu Chang. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding. In *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 510–526, Cham. Springer International Publishing.

Rui Su, Qian Yu, and Dong Xu. 2021. Stvgbert: A visual-linguistic transformer based framework for spatio-temporal video grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1533–1542.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. VL-Adapter: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training Neural Networks with Fixed Sparse Masks. In *Advances in Neural Information Processing Systems*, volume 34, pages 24193–24205. Curran Associates, Inc.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal Few-Shot Learning with Frozen Language Models. In *Advances in Neural Information Processing Systems*, volume 34, pages 200–212. Curran Associates, Inc.

Zhenzhi Wang, Limin Wang, Tao Wu, Tianhao Li, and Gangshan Wu. 2022. Negative Sample Matters: A Renaissance of Metric Learning for Temporal Grounding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(3):2613–2623.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Huijuan Xu, Kun He, L Sigal, S Sclaroff, and K Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*.

Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022a. Tubedetr: Spatio-temporal video grounding with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16442–16453.

Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022b. Zero-Shot Video Question Answering via Frozen Bidirectional Language Models.

Zekun Yang, Noa Garcia, Chenhui Chu, Mayu Otani, Yuta Nakashima, and Haruo Takemura. 2020. BERT representations for Video Question Answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1556–1565.

Runhao Zeng, Haoming Xu, Wenbing Huang, Peihao Chen, Mingkui Tan, and Chuang Gan. 2020. Dense Regression Network for Video Grounding. *arXiv:2004.03545 [cs]*.

Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. 2020. Span-based Localizing Network for Natural Language Video Localization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6543–6554, Online. Association for Computational Linguistics.

Mingxing Zhang, Yang Yang, Xinghan Chen, Yanli Ji, Xing Xu, Jingjing Li, and Heng Tao Shen. 2021a. Multi-Stage Aggregated Transformer Network for Temporal Language Localization in Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12669–12678.

Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. 2021b. Tip-Adapter: Training-free CLIP-Adapter for Better Vision-Language Modeling.

Songyang Zhang, Houwen Peng, Jianlong Fu, Yijuan Lu, and Jiebo Luo. 2021c. Multi-Scale 2D Temporal Adjacency Networks for Moment Localization with Natural Language. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Yang Zhao, Zhou Zhao, Zhu Zhang, and Zhijie Lin. 2021. Cascaded Prediction Network via Segment Tree for Temporal Video Grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4197–4206.

Minghang Zheng, Yanjie Huang, Qingchao Chen, Yuxin Peng, and Yang Liu. 2022. Weakly Supervised Temporal Sentence Grounding With Gaussian-Based Contrastive Proposal Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15555–15564.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision*, 130(9):2337–2348.

Luowei Zhou, Nathan Louis, and Jason J. Corso. 2018a. Weakly-Supervised Video Object Grounding from Text by Loss Weighting and Object Interaction. *arXiv:1805.02834 [cs]*.

Luowei Zhou, Chenliang Xu, and Jason J. Corso. 2018b. Towards Automatic Learning of Procedures From Web Instructional Videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

## A Detailed Results with BERT

We report the detailed results with BERT for ExCL, TMLGA, and DORi in Tables 4, 5 and 6, respectively. We also report the average runtime for training and inference of each TVG model when combined with BERT and the selected adapters in Table 7. However, we note that these values also include the loading time of the samples, which varied accordind to the usage of the cluster during each experiment.

### A.1 Hyper-parameters

To reproduce the results of the original models, we started by using the hyper-parameters reported in the respective papers, achieving close to reported performance for ExCL and TMLGA. Reproducing DORi results was slightly more challenging, where we had to experiment with different weight decays, batch sizes and steps for the learning scheduler.

When adding the PLMs and the adapters, we first started with the same set of hyper-parameters of the original model. In general, it was necessary to slightly change them for the model to properly learn. The hyper-parameters used to train ExCL are specified in Table 8 , Table 9, and Table 10 for Charades-STA, ActivityNet Captions and YouCookII datasets, respectively; to train TMLGA, in Table 11, Table 12, and Table 13, respectively; and to train DORi, in Table 14, Table 15, and Table 16, respectively.

### A.2 Notes on training

Specifically for the fine-tuning of BERT with TMLGA on the Charades-STA dataset, we report the results obtained by applying the linear warm-up to all parameters, including non-BERT ones, as this strategy led to the best results. Moreover, when training DORi with BERT and the PARALLEL adapter on the Charades-STA dataset, we tested different weight decays, but the gradient exploded in all cases. The reported results were obtained with a weight decay of $1e-5$, the best result before the gradient exploded. Finally, we could not find a proper hyper-parameter combination so that the models could learn on the YouCookII with the PARALLEL adapter.

## B Detailed Results with RoBERTa and DeBERTa

Detailed results with RoBERTa and DeBERTa with TMLGA on the Charades-STA and ActivityNet Captions datasets can be found on table 17. This tables is an expansion of the results shown in table 2 in the main text. We did not include results with PREFIX adapters on DeBERTa due to an implementation error on the adapter-transformers library[3]. Furthermore, we also note that in our experiments with the PARALLEL adapter, all the models saturated and reached the same performance on ActivityNet, without properly learning.

### B.1 Hyper-parameters

We report the hyper-parameters used to train TMLGA with RoBERTa on Charades-STA and ActivityNet Captions on Table 18 and Table 19, respectively. We also report the hyper-parameters used to train TMLGA with DeBERTa on both datasets on Table 20 and Table 21.

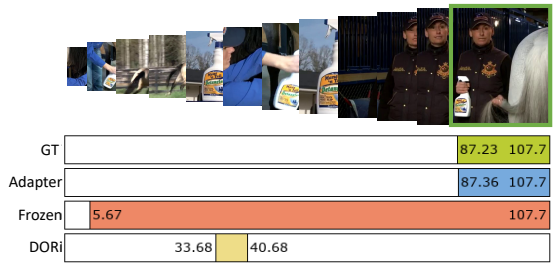
### B.2 Hyper-parameters for best result

Our best results on the Charades-STA dataset was achieved by training DORi with DeBerta, using the PFEIFFER adapter, while the best results on the ActivityNet Captions was achieved by training DORi with DeBERTa, using the INVERSE adapter. The hyper-parameters used to achieve these results can be found in Table 22.

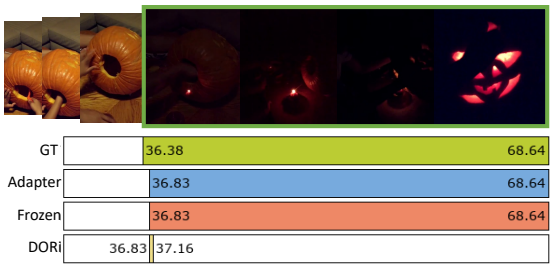
## C Qualitative Results

Finally, we provide a few examples of failure and success of each analyzed TVG model along with BERT. Figure 5 shows examples for DORi and Figure 6 shows examples for ExCL.

[3]Version 3.1.0a0 of the adapter-transformers library.

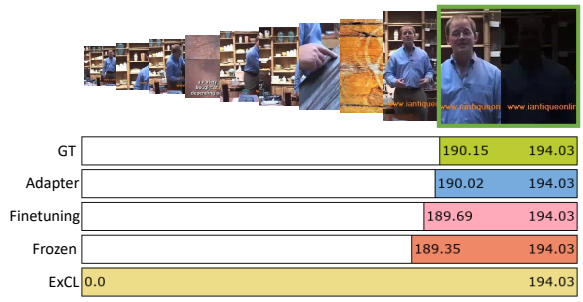



Query: she holds up the product and grabs the horses tail.

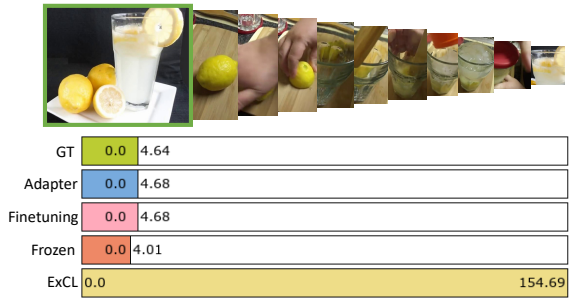



Query: The room turns dark and the two put the pumpkin over a candle and the pumpkin lights up with a carved pumpkin face with a hat.

Figure 5: Examples showing the effects of BERT and adapters with DORi on the ActivityNet. The top image shows an example of a significant performance improvement only when using adapters. On the other hand, the bottom image shows an example where the frozen PLM was sufficient to correctly identify the video segment represented by the query.




Query: We see the man talk and the screen fades to black.




Query: A lemonade drink is displayed on a counter.

Figure 6: Examples showing the effects of BERT and adapters with ExCL on the ActivityNet. Both images show the significant impact of using PLMs on this model's performance, drastically improving results when incorporating frozen BERT. However, while in the top image, the best results were achieved by training using adapters, in the bottom image, we can see that the frozen PLM was sufficient to solve the respective query.

| Method | Params. | Charades-STA | | | | ActivityNet | | | | YouCookII | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU |
| ExCL (ours) | 6.9M | 62.28 | **39.74** | 22.53 | 42.28 | 55.49 | 39.33 | 23.04 | 40.32 | 26.58 | 15.72 | 8.19 | 18.99 |
| + BERT ❄ | 6.9M | **62.93** | 38.44 | 22.23 | **42.38** | 57.21 | 39.66 | 23.79 | 41.45 | 26.63 | 16.15 | 8.51 | 18.87 |
| + HOULSBY | 8.7M | 60.08 | 36.26 | 20.83 | 40.22 | 57.79 | 39.77 | 23.80 | 41.95 | 25.77 | 15.12 | 8.36 | 18.52 |
| + PFEIFFER | 7.8M | 61.59 | 37.15 | 21.51 | 41.06 | **59.35** | 41.27 | 24.86 | **42.83** | 27.18 | 16.15 | 8.88 | 19.46 |
| + INVERSE | 8.1M | 60.81 | 37.69 | 21.10 | 40.85 | 57.58 | 40.34 | 24.79 | 42.06 | 26.72 | 16.44 | 8.79 | 19.43 |
| + PREFIX | 16.8M | 60.54 | 35.24 | 20.22 | 40.98 | 57.49 | 39.55 | 24.10 | 41.77 | 4.93 | 1.83 | 0.66 | 0.06 |
| + COMPACTER | 7.0M | 59.19 | 35.38 | 20.97 | 39.99 | 57.82 | 39.31 | 23.86 | 41.89 | 27.81 | 16.67 | **9.08** | 19.57 |
| + LORA | 7.2M | 60.94 | 36.32 | 21.48 | 40.94 | 57.94 | 40.64 | 25.05 | 42.49 | **28.47** | **17.75** | 9.02 | 19.89 |
| + PARALLEL | 14.0M | 48.52 | 16.61 | 9.44 | 33.77 | 58.60 | 40.99 | 24.73 | 42.26 | 6.41 | 2.58 | 1.12 | 0.06 |
| + Fine-tuning | 116M | 61.75 | 38.36 | **23.44** | 42.00 | 59.10 | **41.83** | **25.42** | 42.36 | 28.18 | 16.84 | 9.02 | **20.08** |

Table 4: Detailed results for ExCL using BERT. Underlined results indicate the best adapter performance, while results in bold indicate the best performance within the dataset.

| Method | Params. | Charades-STA | | | | ActivityNet | | | | YouCookII | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU |
| TMLGA (ours) | 4.7M | 69.49 | 49.97 | 32.72 | 48.29 | 50.84 | 31.13 | 17.86 | 36.90 | 34.42 | 21.99 | 10.94 | 23.63 |
| + BERT ❄ | 4.7M | 70.08 | 49.92 | 31.42 | 48.34 | 52.10 | 32.57 | 18.64 | 37.63 | 34.77 | **23.05** | **12.49** | 24.42 |
| + HOULSBY | 8.7M | 70.97 | 51.69 | 33.84 | 49.31 | **53.98** | 34.13 | 19.48 | 38.57 | **36.08** | 22.77 | **12.49** | **25.19** |
| + PFEIFFER | 7.8M | **71.64** | 51.59 | 33.41 | 49.50 | **53.98** | **35.20** | **20.43** | **38.88** | 34.31 | 21.76 | 11.31 | 23.54 |
| + INVERSE | 8.1M | 71.02 | 52.77 | **34.49** | 49.33 | 52.47 | 33.76 | 19.93 | 37.93 | 35.40 | 22.05 | 11.14 | 24.44 |
| + PREFIX | 16.8M | 71.40 | **52.53** | 33.82 | 49.57 | 53.61 | 34.03 | 19.89 | 38.34 | 18.36 | 10.17 | 4.64 | 13.62 |
| + COMPACTER | 7.0M | 70.27 | 49.73 | 31.37 | 48.31 | 52.15 | 33.85 | 19.62 | 37.78 | 35.34 | 22.25 | 11.05 | 24.13 |
| + LORA | 7.2M | 70.94 | 50.24 | 32.15 | 48.81 | 51.90 | 32.81 | 18.77 | 37.37 | 35.88 | 22.65 | 11.91 | 24.51 |
| + PARALLEL | 14.0M | 70.48 | 51.64 | 33.66 | 49.33 | 43.50 | 23.20 | 11.59 | 32.29 | 5.98 | 2.36 | 1.00 | 0.06 |
| + Fine-tuning | 114M | 71.02 | **52.53** | 33.52 | **49.80** | 53.59 | 34.05 | 19.51 | 37.92 | 35.34 | 21.85 | 11.63 | 24.82 |

Table 5: Detailed results for TMLGA using BERT. Underlined results indicate the best adapter performance, while results in bold indicate the best performance within the dataset.

| Method | Params. | Charades-STA | | | | ActivityNet | | | | YouCookII | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU |
| DORi (ours.) | 10.4M | 72.26 | 57.18 | 40.62 | 53.01 | 57.38 | 40.00 | 24.84 | 41.97 | 43.33 | 29.15 | 17.61 | 30.17 |
| + BERT ❄ | 10.4M | 71.83 | 57.15 | 39.22 | 52.49 | 58.86 | 40.86 | 25.50 | 42.97 | 42.27 | 29.90 | 18.38 | 29.92 |
| + HOULSBY | 8.7M | 72.72 | 57.58 | 40.59 | 53.16 | 60.71 | 43.18 | 26.97 | 43.93 | **46.79** | **32.56** | **19.87** | **32.48** |
| + PFEIFFER | 7.8M | 72.28 | 58.49 | 40.89 | 53.13 | 60.67 | **43.53** | 27.33 | 44.30 | 44.96 | 31.90 | 19.39 | 31.48 |
| + INVERSE | 8.1M | 72.50 | **58.63** | **40.97** | **53.29** | **61.01** | 43.90 | 27.68 | 44.32 | 45.50 | 30.76 | 19.13 | 31.48 |
| + PREFIX | 16.8M | 71.99 | 57.69 | 40.67 | 52.94 | 60.81 | 43.49 | **27.86** | **44.55** | 45.59 | 31.90 | 19.44 | 31.53 |
| + COMPACTER | 7.0M | **72.63** | 57.98 | 40.83 | 52.93 | 60.73 | 43.03 | 27.40 | 44.31 | 43.81 | 30.13 | 18.36 | 30.58 |
| + LORA | 7.2M | 70.73 | 57.31 | 39.76 | 51.89 | 60.90 | 43.34 | 27.46 | 44.42 | 45.42 | 31.44 | 19.47 | 31.56 |
| + PARALLEL | 14.0M | 42.18 | 9.65 | 5.08 | 30.15 | 52.94 | 35.23 | 21.92 | 39.00 | 5.01 | 1.75 | 0.66 | 0.06 |
| + Fine-tuning | 120M | OOM | | | | OOM | | | | OOM | | | |

Table 6: Detailed results for DORi using BERT. Underlined results indicate the best adapter performance, while results in bold indicate the best performance within the dataset.

| Method | Charades-STA | | ActivityNet | | YouCookII | |
|---|---|---|---|---|---|---|
| | Train | Inference | Train | Inference | Train | Inference |
| ExCL (ours) | 33.40 ± 0.88 | 10.70 ± 0.97 | 814.00 ± 79.44 | 425.00 ± 551.66 | 170.37 ± 0.92 | 63.85 ± 61.79 |
| + BERT ❄ | 23.91 ± 1.97 | 7.83 ± 0.38 | 829.40 ± 39.20 | 151.80 ± 7.39 | 169.53 ± 1.01 | 39.44 ± 0.62 |
| + Adapter | 32.18 ± 1.32 | 10.00 ± 0.47 | 833.00 ± 25.31 | 102.85 ± 11.52 | 94.33 ± 0.88 | 21.81 ± 0.60 |
| + Fine-tuning | 45.40 ± 2.67 | 10.10 ± 0.78 | 1883.00 ± 72.12 | 180.00 ± 1.41 | 306.18 ± 66.19 | 36.27 ± 1.73 |
| TMLGA (ours) | 25.04 ± 0.74 | 15.44 ± 0.82 | 884.62 ± 10.56 | 123.75 ± 7.36 | 140.90 ± 2.02 | 43.20 ± 1.470 |
| + BERT ❄ | 25.50 ± 0.85 | 15.93 ± 0.73 | 454.18 ± 2.40 | 82.00 ± 1.00 | 163.27 ± 9.66 | 50.72 ± 47.19 |
| + Adapter | 29.65 ± 0.77 | 17.95 ± 0.78 | 658.19 ± 24.41 | 136.44 ± 4.24 | 166.58 ± 10.26 | 49.33 ± 47.13 |
| + Fine-tuning | 34.89 ± 0.93 | 16.55 ± 0.72 | 768.00 ± 219.64 | 137.92 ± 28.16 | 172.46 ± 1.51 | 37.61 ± 0.87 |
| DORi (ours.) | 631.24 ± 1.13 | 138.20 ± 0.75 | 6704.00 ± 1.41 | 1040.00 ± 4.24 | 2101.00 ± 81.99 | 845.88 ± 1054.53 |
| + BERT ❄ | 607.83 ± 1.52 | 134.67 ± 0.72 | 6599.00 ± 21.66 | 1357.33 ± 23.46 | 2221.50 ± 72.83 | 667.50 ± 1593.11 |
| + Adapter | 633.14 ± 1.34 | 130.14 ± 0.69 | 7415.00 ± 123.30 | 1405.75 ± 34.35 | 2279.50 ± 15.93 | 403.25 ± 29.06 |

Table 7: Average runtime in seconds for training and evaluating each selected TVG model when combining with BERT and adapters.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | Finetuning | 64 |
| | Others | 32 |
| Base LR | Rep., Frozen, Finetuning | 1E-03 |
| | Others | 1E-04 |
| Step | Rep. | 6 |
| | Frozen | 5 |
| | Others | - |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.1 |
| # Epochs | | 15 |
| Gamma | All | 1E-02 |
| Weight Decay | | 1E-05 |

Table 8: Hyper-parameters used to train ExCL with BERT on the Charades-STA dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | Reproduction | 32 |
| | Others | 64 |
| Base LR | PFEIFFER, PARALLEL, COMPACTER, PREFIX | 1E-04 |
| | Others | 1E-03 |
| Step | Finetuning | - |
| | Others | 5 |
| Weight Decay | PFEIFFER, PARALLEL, COMPACTER, PREFIX | 1E-04 |
| | Others | 1E-06 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.2 |
| # Epochs | | 15 |
| Gamma | All | 1E-02 |

Table 9: Hyper-parameters used to train ExCL with BERT on the ActivityNet dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | All | 32 |
| Base LR | | 1E-03 |
| Gamma | | 1E-02 |
| Step | LoRa | 8 |
| | Others | - |
| Weight Decay | Finetunning, Frozen | 1E-05 |
| | Others | 1E-04 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.2 |
| # Epochs | | 15 |

Table 10: Hyper-parameters used to train ExCL with BERT on the YouCookII dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | All | 256 |
| Base LR | | 1E-04 |
| Weight Decay | | 1E-05 |
| Gamma | | 1E-02 |
| Step | Finetuning | - |
| | Others | 6 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.1 |
| # Epochs | | 20 |

Table 11: Hyper-parameters used to train TMLGA with BERT on the Charades-STA dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | All | 64 |
| Base LR | | 1E-04 |
| Weight Decay | | 1E-05 |
| Gamma | | 1E-02 |
| Step | Finetuning | - |
| | Others | 5 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.1 |
| # Epochs | | 15 |

Table 12: Hyper-parameters used to train TMLGA with BERT on the ActivityNet dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | All | 64 |
| Base LR | | 1E-03 |
| Step | | 6 |
| Gamma | | 1E-02 |
| Weight Decay | Prefix | 1E-04 |
| | Others | 1E-05 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.2 |
| # Epochs | | 15 |

Table 13: Hyper-parameters used to train TMLGA with BERT on the YouCookII dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch size | All | 5 |
| Base LR | | 1E-04 |
| Gamma | | 1E-02 |
| Weight Decay | Frozen | 1E-04 |
| | Others | 1E-05 |
| Step | PREFIX | 3 |
| | HOULSBY, PFEIFFER, LORA | 4 |
| | INVERSE, COMPACTER | 5 |
| | Frozen | 6 |

Table 14: Hyper-parameters used to train DORi with BERT on the Charades-STA dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch Size | Rep., Frozen | 8 |
| | Others | 4 |
| Base LR | All | 1E-04 |
| Gamma | | 1E-02 |
| Weight Decay | Rep., Frozen | 1E-04 |
| | Others | 1E-05 |
| Step | Rep. | 3 |
| | All adapters | 4 |
| | Frozen | 6 |

Table 15: Hyper-parameters used to train DORi with BERT on the ActivityNet Captions dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch Size | Rep. | 2 |
| | Others | 4 |
| Base LR | All | 1E-04 |
| Step | | 6 |
| Weight Decay | Rep. | 1E-05 |
| | Others | 1E-04 |
| Gamma | Rep. | 1E-02 |
| | Others | 1E-01 |

Table 16: Hyper-parameters used to train DORi with BERT on the YouCookII dataset.

| Model | Charades-STA | | | | ActivityNet | | | |
|---|---|---|---|---|---|---|---|---|
| | R@0.3 | R@0.5 | R@0.7 | mIoU | R@0.3 | R@0.5 | R@0.7 | mIoU |
| TMLGA (orig.) | 67.53 | 52.02 | 33.74 | 48.22 | 51.28 | 33.04 | 19.26 | 37.78 |
| TMLGA (ours) | 69.49 | 49.97 | 32.72 | 48.29 | 50.84 | 31.13 | 17.86 | 36.90 |
| + BERT ❄ | 70.08 | 49.92 | 31.42 | 48.34 | 52.10 | 32.57 | 18.64 | 37.63 |
| + PFEIFFER | 71.64 | 51.59 | 33.41 | 49.50 | 53.98 | 35.20 | 20.43 | 38.88 |
| + HOULSBY | 70.97 | 51.69 | 33.84 | 49.31 | 53.98 | 34.13 | 19.48 | 38.57 |
| + PREFIX | 71.40 | 52.53 | 33.82 | 49.57 | 53.61 | 34.03 | 19.89 | 38.34 |
| + INVERSE | 71.02 | 52.77 | 34.49 | 49.33 | 52.47 | 33.76 | 19.93 | 37.93 |
| + COMPACTER | 70.27 | 49.73 | 31.37 | 48.31 | 52.15 | 33.85 | 19.62 | 37.78 |
| + LORA | 70.94 | 50.24 | 32.15 | 48.81 | 51.90 | 32.81 | 18.77 | 37.37 |
| + PARALLEL | 70.48 | 51.64 | 33.66 | 49.33 | 43.50 | 23.20 | 11.59 | 32.29 |
| + Fine-tuning | 71.02 | 52.53 | 33.52 | 49.80 | 53.59 | 34.05 | 19.51 | 37.92 |
| + RoBERTa ❄ | 69.73 | 51.34 | 33.49 | 48.91 | 52.58 | 33.8 | 19.62 | 37.89 |
| + PFEIFFER | 71.72 | **53.84** | **34.78** | **49.91** | 54.51 | 35.27 | 20.26 | 38.77 |
| + HOULSBY | 71.08 | 52.98 | 34.19 | 49.28 | 53.56 | 34.34 | 20.16 | 38.89 |
| + PREFIX | **72.28** | 52.42 | 33.98 | 49.90 | 53.08 | 33.19 | 19.48 | 38.47 |
| + INVERSE | 71.53 | 52.69 | 33.98 | 49.50 | 54.36 | 34.85 | 20.46 | 39.35 |
| + COMPACTER | 71.21 | 51.56 | 33.17 | 49.20 | 52.88 | 33.16 | 19.35 | 38.09 |
| + LORA | 71.64 | 51.88 | 33.17 | 49.55 | 53.73 | 34.00 | 19.43 | 38.64 |
| + PARALLEL | 70.99 | 52.93 | 34.01 | 49.19 | 43.50 | 23.20 | 11.59 | 32.29 |
| + Fine-tuning | 71.61 | 53.15 | 33.33 | 49.77 | 52.70 | 33.21 | 19.69 | 38.47 |
| + DeBERTa ❄ | 70.73 | 52.53 | 33.49 | 49.32 | 53.04 | 33.94 | 20.22 | 38.72 |
| + PFEIFFER | 71.34 | 53.49 | 34.65 | 49.78 | 54.37 | 34.70 | 20.49 | 39.30 |
| + HOULSBY | 70.83 | 52.10 | 33.74 | 49.48 | 53.25 | 33.93 | 19.55 | 38.45 |
| + INVERSE | 71.64 | 52.58 | 33.95 | 49.66 | **55.09** | **35.45** | **20.66** | **39.71** |
| + COMPACTER | 70.08 | 51.64 | 32.98 | 48.78 | 53.60 | 34.21 | 20.14 | 38.79 |
| + LORA | 71.18 | 52.34 | 33.31 | 49.18 | 53.98 | 34.53 | 20.02 | 39.00 |
| + PARALLEL* | 70.73 | 53.36 | 34.76 | 49.35 | 43.50 | 23.20 | 11.59 | 32.29 |
| + Fine-tuning | 71.59 | 53.44 | 33.44 | 49.63 | 53.54 | 33.78 | 20.12 | 38.92 |

Table 17: Detailed results combining the TMLGA model with our three pre-trained language encoders and adapters, tested on Charades-STA and ActivityNet Captions. Underlined results indicate the best performance within the model and dataset combination, while the results in bold indicate the best performance within the dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch Size | Finetuning | 256 |
| | Others | 64 |
| Base LR | All | 1E-04 |
| Weight Decay | | 1E-05 |
| Gamma | | 1E-02 |
| Step | | 6 |
| RoBERTa LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.3 |
| # Epochs | | 10 |

Table 18: Hyper-parameters used to train TMLGA with RoBERTa on the Charades-STA dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch Size | All | 64 |
| Base LR | | 1E-04 |
| Weight Decay | | 1E-05 |
| Gamma | | 1E-02 |
| Step | | 6 |
| RoBERTa LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.2 |
| # Epochs | | 10 |

Table 19: Hyper-parameters used to train TMLGA with RoBERTa on the ActivityNet Captions dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch Size | Finetuning | 256 |
| | Others | 64 |
| Base LR | All | 1E-04 |
| Weight Decay | | 1E-05 |
| Gamma | | 1E-02 |
| Step | HOULSBY | 4 |
| | INVERSE | 5 |
| | Frozen, PFEIFFER, PARALLEL, LORA | 6 |
| | COMPACTER | 7 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.3 |
| # Epochs | | 10 |

Table 20: Hyper-parameters used to train TMLGA with DeBERTa on the Charades-STA dataset.

| Hyper-parameter | Method | Value |
|---|---|---|
| Batch Size | All | 64 |
| Base LR | | 1E-04 |
| Weight Decay | | 1E-05 |
| Gamma | | 1E-02 |
| Step | INVERSE | 5 |
| | Others | 6 |
| BERT LR | Finetuning | 1E-04 |
| Warm-up Rate | | 0.2 |
| # Epochs | | 10 |

Table 21: Hyper-parameters used to train TMLGA with DeBERTa on the ActivityNet Captions dataset.

| Dataset | Method | Hyper-parameter | Value |
|---|---|---|---|
| Charades-STA | Pfeiffer | Batch Size | 5 |
| | | Base LR | 1E-04 |
| | | Weight Decay | 1E-04 |
| | | Gamma | 5 |
| | | Step | 1E-02 |
| ANet | Inverse | Batch Size | 4 |
| | | Base LR | 1E-04 |
| | | Weight Decay | 1E-05 |
| | | Gamma | 4 |
| | | Step | 1E-02 |

Table 22: Hyper-parameters used to obtain the best results for DORi with DeBERTa on the Charades-STA and ActivityNet Captions datasets.

## ACL 2023 Responsible NLP Checklist

### A For every submission:

☑ A1. Did you describe the limitations of your work?
*7 (Limitations)*

☐ A2. Did you discuss any potential risks of your work?
*Not applicable. Left blank.*

☑ A3. Do the abstract and introduction summarize the paper's main claims?
*1*

☒ A4. Have you used AI writing assistants when working on this paper?
*Left blank.*

### B ☑ Did you use or create scientific artifacts?

*4.1 and 4.2*

☑ B1. Did you cite the creators of artifacts you used?
*4*

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*8 (Ethics Statement)*

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
*Not applicable. Left blank.*

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
*Not applicable. Left blank.*

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*4.1 and 4.2*

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
*4.1*

### C ☑ Did you run computational experiments?

*4*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*4.2, 7*

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
*4.2 and Appendices A and B*

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
*4.2 and Appendices A and B*

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
*4.2*

**D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
*No response.*

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
*No response.*

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
*No response.*

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
*No response.*

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
*No response.*