Parameter-efficient Tuning of Pretrained Visual-Language Models in Multitask Robot Learning

Marcel Mittenbuehler[1], **Ahmed Hendawy**[1,2], **Carlo D'Eramo**[2,3], **Georgia Chalvatzaki**[1,2]

[1]Technische Universität Darmstadt, Germany

[2]Hessian.AI, Germany [3]University of Würzburg, Germany

marcel.mittenbuehler@stud.tu-darmstadt.de

Abstract: Multimodal pretrained visual-language models (pVLMs) showcased excellence across several applications, like visual question-answering. Their recent use for policy learning manifested promising avenues for augmenting robotic capabilities in the real world. This paper delves into the problem of parameter-efficient tuning of pVLMs for adapting them to robotic manipulation tasks with low-resource data. We showcase how Low-Rank Adapters (LoRA) can be injected into behavioral cloning temporal transformers to fuse language, multi-view images, and proprioception for multitask robot learning, even for long-horizon tasks. Preliminary results indicate our approach vastly outperforms baseline architectures and tuning methods, paving the way toward parameter-efficient adaptation of pretrained large multimodal transformers for robot learning with only a handful of demonstrations.

1 Introduction

Multimodal pretrained models, e.g., pretrained visual-language models (pVLM) like VisualBERT [17], CLIP [25], PALM-e [8], demonstrated state-of-the-art performance in various application domains [1, 18, 36, 16]. A large host of research works is also dedicated to investigating the zero- or few-shot performance of such pretrained models in new tasks [37, 21, 30] or even their efficient adaptation in light of low-resource task-specific data [38, 10, 39, 33, 40]. It was in the last couple of years that robotics research has also explored the application of pVLMs in robotic policy learning, mainly through large-scale training and finetuning of large models to a suite of robotic navigation [28, 13, 31] and manipulation tasks [31, 23, 14, 29], showcasing encouraging results [2, 3, 5].

Multimodal information alleviates the need for complicated engineered goal specifications in robotic tasks [35, 6], and, additionally, they offer an excellent opportunity for intuitive interaction with humans, e.g., through natural language and images [4, 9]. Further, multimodal information is very beneficial for learning robot policies, thanks also to the capacity of the Transformers architecture [34], as it allows the effective combination of symbolic and geometric information for computing robot control actions. Notable efforts in the robot learning community have been oriented toward the collection of large-scale datasets for training robotic pVLMs of billion parameters, demonstrating the ability of such models to adapt to new domains [8, 5]. Yet, a question emerges naturally when it comes to million or even billion parameter robotic pVLMs; *how can we build on top of those pretrained models to quickly adapt to new, unseen tasks with a limited amount of data, and without requiring necessarily massive compute power?*

To this end, this paper presents preliminary evidence on the efficacy of parameter-efficient adaptation of pVLMs [11, 24] into robotic domains and tasks, given a small number of demonstrations. Particularly, we showcase how we can exploit Low-Rank Adapters (LoRA) [12] in CLIP frozen visual and text encoders within a behavioral cloning temporal transformer architecture [19, 27] for multitask robot policy learning, using natural language goal specifications, multi-view images, and proprioceptive information, to learn continuous robot actions for a variety of manipulation tasks.

Our preliminary results suggest that, with a low-data setting and given an expressive pretrained frozen model, we can significantly outperform both dedicated architectures trained from scratch [19] by on average 15 % higher success rates, even finetuned versions of our proposed models (without adapters)

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

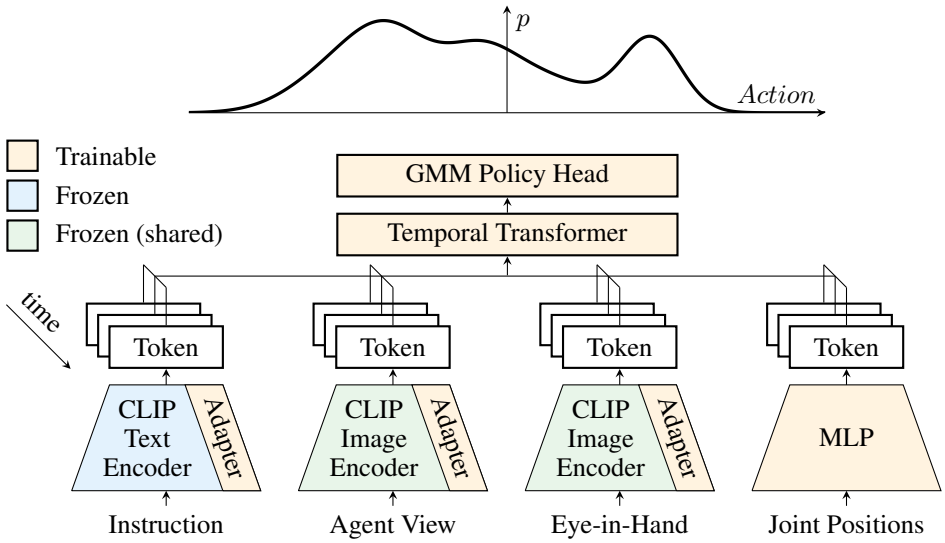

Figure 1: Schematic overview of our model. In the first stage, it encodes inputs for multiple timesteps into tokens. These are then passed to the temporal transformer, which passes the last token to the GMM policy head that outputs a GMM. The CLIP model is frozen during training, and the image encoder is shared between two image streams with different adapters.

by 21 % higher success rates. Our results show a promising avenue for parameter-efficient adaptation of pretrained large multimodal transformers for robot learning, which can be adapted only with a handful of demonstrations, paving the way for more democratized usage of such methods in robotics.

2 Approach

The architecture for our approach is similar to the ResNet-T in [19] architecture. It consists of a task embedding by a pre-trained language transformer, two image encoders for two separate camera streams, and an extra modality encoder to pass in joint positions. The output tokens of multiple timesteps are passed to a temporal transformer, whose last output token is processed by the Gaussian mixture model (GMM) policy head to output a distribution of possible actions. The architecture is shown in Figure 1.

We use a multimodal model for the text and image encoders to utilize the rich embedding information of a pre-trained transformer network. Finetuning two image encoders and one text model can be problematic, considering the small number of examples commonly available in robotics and the strong correlations between temporal proximate observations. Instead, we propose to insert adapters into the pre-trained model, with separate adapters for each image stream. The advantage of adapters is two-fold: first, they require fewer parameters to tune and should, therefore, be more resilient with few demonstrations available; second, they allow us to reuse one transformer model for both image streams, lowering the memory requirements during training. Moreover, this approach is scalable to more camera streams on the robot without much overhead.
Adapters are small, trainable models inserted at different points – depending on the adapter type – of a transformer head that allows parameter-efficient tuning of a large pre-trained transformer model. For example, a bottleneck adapter [11] is inserted in parallel with the feed-forward layer. This work instead explores the application of low-rank adaptation (LoRA) [12], which modifies the attention weight matrices in the attention layer as shown in Figure 2. Let d be the latent dimension, $W \in \mathbb{R}^{d \times d}$ one of the weight matrices, x its input, and h its output, then LoRA introduces two other matrices $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$ with lower rank r such that $r \leq d$. The low-rank approximation is performed by creating an auxiliary term $\delta W = BA$ with the new output as $h = Wx + \delta Wx$. If δW is zero, the output is that of the pre-trained model. During training, changing the parameters of A and B results in an adjustment of the model. As $r \ll d$ is usually chosen, this is more parameter-efficient than updating the whole matrix. Compared to other adapter strategies, the advantage is its speed during inference as LoRA can be integrated into the weight matrices of the model. Therefore, we

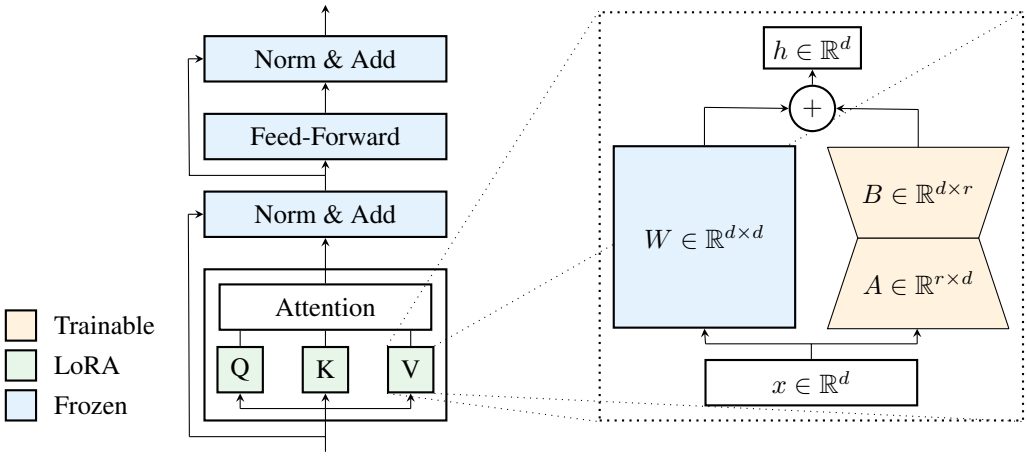

Figure 2: Transformer architecture with LoRA adapters applied to the attention weight matrices.

can benefit from a lower memory footprint on the backward pass during training whilst having a fast model for inference. The latter part is particularly important for achieving real-time control. Other adapters such as [20, 11, 24] likely also achieve good training performance; however, they usually introduce additional overhead.

We choose CLIP [25] for the pre-trained multimodal model, as the image and text encoders are trained to output tokens into one combined embedding. Moreover, it demonstrated generalization capabilities in zero-shot and transfer-learning experiments, making it a good fit for our purposes. Other models, such as [26, 17, 8], may also be considered.

3 Evaluation

We evaluate our approach on the main Libero benchmark suites [19] containing ten tasks each (excluding Libero 90) in a multitask setup (see Appendix A for details). Each model is trained for ten epochs and evaluated after each epoch in simulation, measuring the success rate. Below, we report the highest success rate of an agent during these ten epochs for three random seeds. For better comparability, all models use the same training hyperparameters. Baseline model parameters remain unchanged, and our model utilizes the same parameters as the ResNet-T baseline. All parameters are listed in Appendix C.

The evaluation is split into two parts: First, we compare our approach with the top-performing baseline models in Libero; Then, we perform ablations by replacing the adapters in the CLIP model with finetuning and training from scratch to gain insights into the performance differences.

3.1 Experimental Setup

Libero [19] is a task suite of language-conditioned manipulation tasks with sparse rewards and human demonstrations for behavioral cloning [32]. It is intended as a benchmark for continual learning in robotics by providing multiple tasks that a robot should learn. More precisely, it has four task suites consisting of ten tasks and one containing 90 for pretraining. Moreover, it includes baseline models tested on the datasets in continual and multi-task settings. Evaluation is performed in simulation with randomized initial parameters. The dataset is explained with additional details in Appendix A.

3.2 Baseline Comparison

Libero provides baseline models to evaluate performance on the datasets. We chose to compare against their most performant models, the ResNet-T and ViLT-T. The main difference to our model is that a ResNet or a ViLT [15] encodes the images. A pre-trained Bert model [7] initially embeds the text that is then transformed by a trainable linear layer. Moreover, the temporal transformer of the ViLT-T has a twice as large embedding as our and the ResNet-T model; however, we did not want to change the baseline model parameters for a fair comparison.

Peak success rates, shown in Table 1, show that our CLIP-T (Adapter) model outperforms the

Table 1: Peak success rate achieved by the models within 10 training epochs.

Model	LIBERO-Long	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal
ResNet-T	0.36 ± 0.02	0.77 ± 0.02	0.81 ± 0.07	0.73 ± 0.04
ViLT-T	0.34 ± 0.01	0.69 ± 0.03	0.80 ± 0.03	0.74 ± 0.05
CLIP-T (Finetuning)	0.24 ± 0.06	0.61 ± 0.09	0.81 ± 0.04	0.76 ± 0.04
CLIP-T (Scratch)	0.23 ± 0.10	0.58 ± 0.07	0.83 ± 0.01	0.75 ± 0.05
CLIP-T (Adapter, ours)	$\mathbf{0.63 \pm 0.02}$	$\mathbf{0.87 \pm 0.01}$	$\mathbf{0.90 \pm 0.02}$	$\mathbf{0.85 \pm 0.02}$

Table 2: Success rates for different LoRA ranks

Rank r	LIBERO-Long	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal
1	0.52 ± 0.03	0.85 ± 0.03	0.92 ± 0.04	0.84 ± 0.03
2	0.59 ± 0.04	0.81 ± 0.04	$\mathbf{0.94 \pm 0.03}$	$\mathbf{0.85 \pm 0.05}$
4	0.59 ± 0.02	0.85 ± 0.07	0.93 ± 0.02	$\mathbf{0.85 \pm 0.02}$
8	$\mathbf{0.63 \pm 0.02}$	$\mathbf{0.87 \pm 0.01}$	0.90 ± 0.02	$\mathbf{0.85 \pm 0.02}$

baselines on all datasets with a $14.2\,\%$ higher success rate on average than the ResNet-T and $16.9\,\%$ higher than the ViLT-T. The results on Libero-Long show a much larger difference in success rate than the other models. The task suite consists of ten long-horizon tasks and is the most difficult to learn; nevertheless, the adapter strategy performed almost twice as well as the other models. As the temporal transformer and output GMM remain similar to the baselines, this strongly suggests that the model benefits from the pre-trained encoding of the CLIP model using adapters.

3.3 Ablation Study

A common strategy to adjust pre-trained models to a new task is fine-tuning all parameters. However, a challenge in robotics is that datasets are comparatively small, as humans must perform demonstrations. Moreover, the observations in the camera feed are highly correlated in time. Thus, training a model efficiently on a limited number of samples is critical.
To better understand the effect of this, we replaced our adapters by fine-tuning and scratch-training the CLIP model. As before, the peak success rate in Table 1 shows that adapters outperform training the full model. Moreover, fine-tuning performs very similarly to scratch training, indicating that overfitting of the model may happen, resulting in sub-optimal embeddings. We assume that the lower number of parameters in the adapter model helps to stabilize the training process. We also tested the effect of the rank r of the LoRA adapters. For this, we kept all parameters as in Appendix C, but varied r and set $\alpha = r$ as suggested in [12]. The results are shown in Table 2 with the best results highlighted. A rank $r = 8$ performed best on three out of the four benchmarks. Only on LIBERO-Object, it performed worse than adapters with fewer parameters.

4 Conclusion and Discussion

This paper presented the application of adapters in large pre-trained multi-modal transformer models to achieve parameter-efficient adaptation in the encoding stage of a model. Less tunable parameters result in memory-efficient training; more importantly, it may stabilize training on small or highly correlated datasets, which is common in robot learning with behavioral cloning. Our approach outperformed the baseline models on the Libero benchmark, although we only adjusted the encoding stage of the model. This indicates that the model benefited from the pre-trained encodings. Moreover, the adapters outperformed fine-tuning and training from scratch, with similar results in the case of fine-tuning and scratch training, suggesting that overfitting happens when parameters are not frozen. A lower learning rate or more training epochs may change the results; however, this raises the question of how efficiently a model should be trainable. Adapters demonstrated good performance and efficient learning. Moreover, more examples could help the large models converge at the cost of lower efficiency.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) (Project: 01IS22078), the DFG Emmy Noether Programme No. CH 2676/1-1, and the Hessian.AI through the Connectom Fund on Lifelong Explainable Robot Learning and the project 'The Third Wave of Artificial Intelligence – 3AI' by the Ministry for Science and Arts of the state of Hessen.

References

[1] Jean-Baptiste Alayrac et al. "Flamingo: a visual language model for few-shot learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23716–23736.

[2] Anthony Brohan et al. "Rt-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[3] Anthony Brohan et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).

[4] Arthur Bucker et al. "Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 978–984.

[5] Open X-Embodiment Collaboration et al. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. https://robotics-transformer-x.github.io. 2023.

[6] Yuchen Cui et al. "Can Foundation Models Perform Zero-Shot Task Specification For Robot Manipulation?" In: *Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 893–905.

[7] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[8] Danny Driess et al. "Palm-e: An embodied multimodal language model". In: *arXiv preprint arXiv:2303.03378* (2023).

[9] Yuqing Du et al. "Guiding pretraining in reinforcement learning with large language models". In: *arXiv preprint arXiv:2302.06692* (2023).

[10] Peng Gao et al. "Clip-adapter: Better vision-language models with feature adapters". In: *International Journal of Computer Vision* (2023), pp. 1–15.

[11] Neil Houlsby et al. "Parameter-efficient transfer learning for NLP". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[12] Edward J Hu et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[13] Chenguang Huang et al. "Visual language maps for robot navigation". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 10608–10615.

[14] Yunfan Jiang et al. "Vima: General robot manipulation with multimodal prompts". In: *arXiv preprint arXiv:2210.03094* (2022).

[15] Wonjae Kim, Bokyung Son, and Ildoo Kim. "Vilt: Vision-and-language transformer without convolution or region supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5583–5594.

[16] Kenton Lee et al. "Pix2struct: Screenshot parsing as pretraining for visual language understanding". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 18893–18912.

[17] Liunian Harold Li et al. "Visualbert: A simple and performant baseline for vision and language". In: *arXiv preprint arXiv:1908.03557* (2019).

[18] Zhihong Lin et al. "Medical visual question answering: A survey". In: *Artificial Intelligence in Medicine* (2023), p. 102611.

[19] Bo Liu et al. "LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning". In: *arXiv preprint arXiv:2306.03310* (2023).

[20] Haokun Liu et al. "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 1950–1965.

[21] Minghua Liu et al. "Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 21736–21746.

[22] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[23] Suraj Nair et al. "R3m: A universal visual representation for robot manipulation". In: *arXiv preprint arXiv:2203.12601* (2022).

[24] Jonas Pfeiffer et al. "Mad-x: An adapter-based framework for multi-task cross-lingual transfer". In: *arXiv preprint arXiv:2005.00052* (2020).

[25] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[26] Christoph Schuhmann et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.

[27] Nur Muhammad Shafiullah et al. "Behavior Transformers: Cloning k modes with one stone". In: *Advances in neural information processing systems* 35 (2022), pp. 22955–22968.

[28] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action". In: *Conference on Robot Learning*. PMLR. 2023, pp. 492–504.

[29] Mohit Sharma et al. "Lossless adaptation of pretrained vision models for robotic manipulation". In: *arXiv preprint arXiv:2304.06600* (2023).

[30] Haoyu Song et al. "Clip models are few-shot learners: Empirical studies on vqa and visual entailment". In: *arXiv preprint arXiv:2203.07190* (2022).

[31] Austin Stone et al. "Open-world object manipulation using pre-trained vision-language models". In: *arXiv preprint arXiv:2303.00905* (2023).

[32] Faraz Torabi, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation". In: *arXiv preprint arXiv:1805.01954* (2018).

[33] Uddeshya Upadhyay et al. "ProbVLM: Probabilistic Adapter for Frozen Vison-Language Models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1899–1910.

[34] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[35] Renhao Wang et al. "Programmatically Grounded, Compositionally Generalizable Robotic Manipulation". In: *arXiv preprint arXiv:2304.13826* (2023).

[36] Antoine Yang et al. "Vid2seq: Large-scale pretraining of a visual language model for dense video captioning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 10714–10726.

[37] Zhuolin Yang et al. "Re-vilm: Retrieval-augmented visual language model for zero and few-shot image captioning". In: *arXiv preprint arXiv:2302.04858* (2023).

[38] Chuhan Zhang et al. "Making the Most of What You Have: Adapting Pre-trained Visual Language Models in the Low-data Regime". In: *arXiv preprint arXiv:2305.02297* (2023).

[39] Xinyun Zhang et al. "Towards Versatile and Efficient Visual Knowledge Injection into Pre-trained Language Models with Cross-Modal Adapters". In: *arXiv preprint arXiv:2305.07358* (2023).

[40] Kecheng Zheng et al. "Regularized Mask Tuning: Uncovering Hidden Knowledge in Pre-trained Vision-Language Models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 11663–11673.

A Libero Dataset

Libero introduced multiple datasets to test the ability of an agent in various regards. For a full overview, we recommend reading [19]; however, we will give a brief introduction into the datasets here.

A.1 LIBERO-Spatial

Libero Spatial tests the ability of an agent to generalize one kind of task to different start. Each task consists of picking up a bowl and placing it on a plate; however, the location specifier of the bowl differs. The tasks are of the following structure

- Pick up the black bowl from table center and place it on the plate.
- Pick up the black bowl next to the cookie box and place it on the plate.
- Pick up the black bowl <spatial specifier> and place it on the plate.

A.2 LIBERO-Object

Similar to Libero Spatial, this benchmark expects the agent to perform the same action. Instead of the location specifier differing, the object that should be moved is always different. This allows to test the performance of the vision encoder of a model. The tasks include

- Pick up the alphabet soup and place it in the basket.
- Pick up the cream cheese and place it in the basket.
- Pick up the <object> and place it in the basket.

A.3 LIBERO-Goal

In Libero Goal, the expected goal state in the simulation environment varies. Intuitively, this benchmark requires the agent to learn a range of different tasks. To give some examples

- Put the wine bottle on top of the cabinet.
- Open the top drawer and put the bowl inside.
- Turn on the stove.
- ...

A.4 LIBERO-Long

Finally, Libero Long provides a benchmark to test the long horizon capabilities of a model. The tasks mainly consist of two tasks that have similar complexity such as the previous benchmarks. Moreover, the order in which the tasks is executed matters; for example, when the agent is expected to place something in a drawer and then close it. This benchmark includes tasks like

- Turn on the stove and put the moka pot on it.
- Put the black bowl in the bottom drawer of the cabinet and close it.
- Put the yellow and white mug in the microwave and close it.
- ...

B Performance Metrics

Training of the models was performed on Nvidia RTX 3080 GPUs with $10\,\mathrm{GB}$ VRAM. The latter dictated the batch size for all models as otherwise the scratch training would not have fit into memory. The training times of the models are listed in Table 3 with the number of trainable parameters in Table 4. One argument for the use of LoRA was the possibility of merging the adapters into the pVLM to achieve better performance. Therefore, we also report the inference runtimes in Table 5 with LoRA not merged. In case of a merged LoRA, inference time is the same as for the CLIP-T (Finetuning).

Table 3: Average training time for one epoch with batch size 8

Model	LIBERO-Long	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal
ResNet-T	$(20 \pm 5)\,\mathrm{min}$	$(8 \pm 3)\,\mathrm{min}$	$(11 \pm 4)\,\mathrm{min}$	$(9 \pm 4)\,\mathrm{min}$
ViLT-T	$(31 \pm 4)\,\mathrm{min}$	$(14 \pm 6)\,\mathrm{min}$	$(18 \pm 5)\,\mathrm{min}$	$(14 \pm 4)\,\mathrm{min}$
CLIP-T (Finetuning)	$(114 \pm 10)\,\mathrm{min}$	$(50 \pm 6)\,\mathrm{min}$	$(63 \pm 9)\,\mathrm{min}$	$(55 \pm 6)\,\mathrm{min}$
CLIP-T (Scratch)	$(113 \pm 10)\,\mathrm{min}$	$(54 \pm 9)\,\mathrm{min}$	$(63 \pm 7)\,\mathrm{min}$	$(55 \pm 7)\,\mathrm{min}$
CLIP-T (LoRA, $r = 8$)	$(93 \pm 10)\,\mathrm{min}$	$(41 \pm 6)\,\mathrm{min}$	$(48 \pm 6)\,\mathrm{min}$	$(43 \pm 5)\,\mathrm{min}$

Table 4: Number of trainable parameters for each model

Model	Trainable Parameters
ResNet-T	5.38×10^6
ViLT-T	7.92×10^6
CLIP-T (Finetuning)	154×10^6
CLIP-T (Scratch)	154×10^6
CLIP-T (Adapter, $r = 8$)	3.69×10^6
CLIP-T (Adapter, $r = 4$)	3.29×10^6
CLIP-T (Adapter, $r = 2$)	3.10×10^6
CLIP-T (Adapter, $r = 1$)	3.00×10^6

Table 5: Average inference time for one sample

Model	Time in ms
ResNet-T	5.4 ± 0.4
ViLT-T	7 ± 1
CLIP-T (Finetuning)	25 ± 1
CLIP-T (Scratch)	24 ± 2
CLIP-T (Adapter, $r = 8$)	31 ± 1

C Hyperparameters

All our experiments used the same training hyperparameters with a learning rate of 1×10^{-4}, a batch size of 8, and using the AdamW optimizer [22]. The model parameters for the ResNet-T are given in Table 6, for the ViLT-T in Table 7, and for our approach in Table 8.

Table 6: Model parameters for the ResNet-T model

Parameter	Value
extra_info_hidden_size	128
img_embed_size	64
transformer_num_layers	4
transformer_num_heads	6
transformer_head_output_size	64
transformer_mlp_hidden_size	256
transformer_dropout	0.1
transformer_max_seq_len	10

Table 7: Model parameters for the ViLT-T model

Parameter	Value
extra_info_hidden_size	128
img_embed_size	128
spatial_transformer_num_layers	7
spatial_transformer_num_heads	8
spatial_transformer_head_output_size	120
spatial_transformer_mlp_hidden_size	256
spatial_transformer_dropout	0.1
spatial_down_sample_embed_size	64
temporal_transformer_num_layers	4
temporal_transformer_num_heads	6
temporal_transformer_head_output_size	64
temporal_transformer_mlp_hidden_size	256
temporal_transformer_dropout	0.1
temporal_transformer_max_seq_len	10

Table 8: Model parameters for the CLIP-T model (ours)

Parameter	Value
extra_info_hidden_size	128
img_embed_size	64
transformer_num_layers	4
transformer_num_heads	6
transformer_head_output_size	64
transformer_mlp_hidden_size	256
transformer_dropout	0.1
transformer_max_seq_len	10
lora_r	8
lora_alpha	8